

TK 2000 *III*

manual técnico

MICRODIGITAL

Prezado usuário,

Este Manual Técnico do TK-2000 COLOR representa um significativo avanço tecnológico para quem está aprendendo a conhecer o equipamento.

O objetivo deste manual é permitir que pessoas que se interessam em aproveitar mais eficientemente seu TK-2000 COLOR, tenham suficiente informação para fazê-lo. Sem dúvida, as informações nele contidas serão de grande utilidade para aqueles que não se satisfazem só com o poderoso BASIC do TK-2000, mas desejam se aprofundar ainda mais no uso dos recursos que oferece a programação em linguagem de máquina.

Serão explicadas as características de comunicação entre o computador e o usuário; uso do sistema de monitor-disassembler assim como o mini-assembler; o mapeamento da memória com informações sobre posições úteis, assim como outras informações importantes.

Este é um campo muito fértil para a aplicação da criatividade e inventividade, e a MICRODIGITAL, ao tomar a iniciativa de acompanhar o TK-2000 COLOR com o Manual Técnico, acredita estar situando seus usuários na vanguarda da informática.

Dpto de Suporte ao Usuário

MICRODIGITAL LTDA.

É proibida a reprodução total ou parcial deste manual sem prévia autorização por escrito da MICRODIGITAL ELETRÔNICA LTDA.

INDICE:

CAPITULO I - O SISTEMA MONITOR

- I.1 - Entrando no Monitor 5
- I.2 - Examinando a Memória 5
- I.3 - Alterando o conteúdo da memória 7
- I.4 - Transferindo um bloco de memória 8
- I.5 - Comparando dois blocos de memória 9
- I.6 - Carregando blocos de memória em cassette 10

CAPITULO II - CRIANDO E EXECUTANDO PROGRAMAS EM LINGUAGEM DE MAQUINA

- II.1 - Disassembler 13
- II.2 - Outros comandos 14
- II.3 - Examinando registradores 15

CAPITULO III - O MINI-ASSEMBLER

- III.4 - Mini-Assembler 17
- III.5 - Programas com o Mini-Assembler 17

CAPITULO IV - O TECLADO

- IV.1 Introdução 21
- IV.2 A Rotina de leitura do teclado 23
- IV.3 RESET 25

CAPITULO V - ORGANIZAÇÃO DA MEMORIA

- V.1 - Descrição das páginas de memória 27
- V.2 - Memória RAM 27
- V.3 - Memória ROM 28
- V.4 - I/O 29
- V.5 - Video 29
- V.6 - TK-2000 II Versões 64K e 128K de RAM 30

CAPITULO VI - SOM

- VI.1 - Introdução 32
- VI.2 - Canal de Som 32
- VI.3 - Programando Tonalidades Musicais 33

CAPITULO VII - VIDEO

VII.1 - Modo Texto

- VII.1.1 - Funcionamento do Cursor 35
 - VII.1.2 - Controlando a posição do cursor através do comando PRINT 36
 - VII.1.3 - Os registradores de posição de cursor 37
 - VII.1.4 - Formatos alternativos de TEXTO 39
 - VII.1.5 - Regulando a primeira coluna de TEXTO 39
 - VII.1.6 - Regulando o número de caracteres por linha 40
 - VII.1.7 - Regulando a posição da linha de topo 41
 - VII.1.8 - Regulando a última linha da tela 42
 - VII.1.9 - Programando a janela de TEXTO 42
- VII.2 - Modo de operação em baixa resolução 43
- VII.3 - Modo gráfico de Alta Resolução 45
- VII.3.1 - Introdução 45
 - VII.3.2 - Memória ocupada pela Alta Resolução 45
 - VII.3.3 - Formação de figuras em Alta Resolução 48
 - VII.3.4 - Utilizando cores em gráficos de Alta Res. 54
 - VII.3.5 - Conclusão 63

CAPITULO VIII - O HARDWARE

- VIII.1 - Diagrama de blocos 67
- VIII.2 - O Microprocessador 69
- VIII.3 - Conector de Saída - Expansão 70
- VIII.4 - Interface de cassette - Comando MOTOR 74
- VIII.5 - Conector de Joystick 74
- VIII.6 - Interface de Impressora 75
- VIII.7 - Fonte de Alimentação 77

- APENDICE A - Instruções do Microprocessador 6502 79
- APENDICE B - Diferenças entre o TK-2000 e o Apple II 93
- APENDICE C - Endereçamento das linhas de vídeo 97
- APENDICE D - Subrotinas úteis no TK-2000 107
- APENDICE E - Bibliografia 113
- APENDICE F - Mapa da memória 115
- APENDICE G - TK-2000 64K 117
- APENDICE H - TK-2000 128K 120

Capítulo I - SISTEMA MONITOR

Dentro da ROM do TK-2000 existe um poderoso programa chamado Monitor-Disassembler (MD). Ele atua como supervisor do sistema de operação. Usando-o você pode descobrir os segredos das 65536 posições de memória.

Com ele você pode verificar um, ou vários endereços, pode mudar o conteúdo de qualquer endereço da RAM, pode escrever diretamente programas em linguagem de máquina, pode armazenar e carregar blocos de dados em fita cassette e pode comparar e transferir blocos de dados.

I.1 Entrando no Monitor

O programa MD começa no endereço \$FF61 (decimal 65377 ou -159). Para entrar no MD, deve ser usada inicialmente uma instrução CALL -159 ou simplesmente LM. O MD responderá com uma arroba (@) do lado esquerdo da tela com o cursor a direita dele. O MD aceita linhas de entrada de dados com até 255 caracteres e alguns comandos que são aceitos assim que você pressionar a tecla RETURN.

Para retornar ao BASIC pressione CONTROL-C e RETURN ou simplesmente RESET.

O MD opera diretamente com números hexadecimais, usando 4 dígitos para expressar endereços (os 0 mais significativos, se houverem, não precisam ser digitados) e 2 dígitos para expressar conteúdos. O programa MD possui 5 registradores, dois dos quais são especiais: um dá o endereço da última posição de memória utilizada por você e o outro dá o endereço da próxima posição cujo conteúdo pode ser mudado. Eles são chamados de: "última posição aberta" (UPA) e "próxima posição alterável" (PPA). A utilidade destes dois registradores será demonstrada adiante.

I.2 Examinando a Memória

Quando você digitar só o endereço de uma posição da memória, o MD responde com o endereço digitado, um traço, um espaço e o valor da posição de memória endereçada, por exemplo:

```
@F800                                <RETURN>
@F800- 4C
@-
```

Cada vez que o MD mostra o valor contido numa posição, lembra desta como UPA. Neste caso também é considerada como PPA.

Se você digitar um ponto (.) seguido de um endereço, o

MD mostrará os valores contidos nas posições desde a UPA até o endereço digitado (desde que este último seja maior que o correspondente a UPA). Observe os exemplos abaixo:

```
@FA00                                <RETURN>
```

```
FA00- DB  
@.FA0F                                <RETURN>
```

```
FA01- 62 5A 48 26 62 94 88  
FA08- 54 44 C8 54 68 44 E8 94  
@FB00                                <RETURN>
```

```
F800- 4C  
@.F812                                <RETURN>
```

```
F801- CA EF 4A 08 20 F5 FB  
F808- 20 73 FC A5 09 28 90 02  
F810- 69 0F 85  
@.F82A                                <RETURN>
```

```
F813- 09 60 86 EA EA  
F818- EA 20 00 F8 C4 2C B0 11  
F820- C8 20 C8 EF 90 F6 69 01  
F828- 48 20 00  
@
```

Pode-se observar que a listagem procura iniciar as linhas a partir do endereço cujo dígito menos significativo é 0 ou 8, e apresenta até 8 posições contínuas. A primeira linha imprime as suficientes informações para poder começar a segunda linha no formato padrão.

Outra forma de usar o ponto é imprimindo o endereço inicial, ponto e endereço final da área de memória a ser examinada.

```
@F200.F22F                            <RETURN>
```

```
F200- 00 1C 22 2A 3A 1A 02 3C  
F208- 00 08 14 22 22 3E 22 22  
F210- 00 1E 22 22 1E 22 22 1F  
F218- 00 1C 22 02 02 02 22 1C  
F220- 00 1E 22 22 22 22 22 1E  
F228- 00 3E 02 02 1E 02 02 3E  
@F900.F910                            <RETURN>
```

```
F900- F8 69 BF 20 ED FD CA D0  
F908- EC 20 48 F9 A4 2F A2 06  
F910- E0  
@
```

Outro método de examinar a memória é indicando o endereço inicial e logo pressionando RETURN tantas vezes quanto achar necessário.


```

0F805                                <RETURN>

F805- 20
0                                     <RETURN>
    F5 FB
0                                     <RETURN>
F808- 20 73 FC A5 09 28 90 02
0F832                                <RETURN>

F832- A0
0                                     <RETURN>
    2F D0 02 A0 27
0                                     <RETURN>
F838- 84 2D A0 27 A9 00 85 30
0

```

I.3 Alterando o conteúdo da Memória

Agora veremos como usar a PPA. Escreva a seguinte sequência:

```

0420                                <RETURN>

0420- 00
0:12                                <RETURN>

```

Pronto. O conteúdo da PPA foi alterado com o valor que você digitou. Confira examinando a posição novamente:

```

0420                                <RETURN>

0420- 12
0

```

Você pode alterar o conteúdo de várias posições consecutivas separando os novos valores com um espaço.

```

0400:12 23 34 45 56 67 78 89    <RETURN>

0400                                <RETURN>

0400- 12
0                                     <RETURN>
    23 34 45 56 67 78 89
0420:0 1 2 3                      <RETURN>

0:4 5 6 7                          <RETURN>

0420.427                            <RETURN>

0420- 00 01 02 03 04 05 06 07
0

```


I.4 Transferindo um bloco de memória

Você pode considerar um bloco de memória (definido por dois endereços separados por um ponto) como uma entidade autônoma e movimentar ou transferir de um lugar para outro usando o comando M (MOVE, de movimentar). Deve ser indicado o endereço inicial e final do bloco de memória e ainda o endereço inicial do destino (posição de memória a partir da qual será deslocado o bloco de memória). O comando então terá a seguinte forma:

(destino) < (início) . (final) M

sendo que as palavras entre as chaves devem ser substituídas por endereços hexadecimais. Por exemplo:

```
@460.46F                                <RETURN>

0460- 00 00 00 00 00 00 00 00
0468- 00 00 00 00 00 00 00 00
@400:00 11 22 33 44 55 66 77 88 99 AA BB CC DD
EE FF                                <RETURN>

@400.40F                                <RETURN>

0400- 00 11 22 33 44 55 66 77
0408- 88 99 AA BB CC DD EE FF
@460<400.40FM                          <RETURN>

@460.46F                                <RETURN>

0460- 00 11 22 33 44 55 66 77
0468- 88 99 AA BB CC DD EE FF
@470.477                                <RETURN>

0470- 00 00 00 00 00 00 00 00
@470<400.407M                          <RETURN>

@470.477                                <RETURN>

0470- 00 11 22 33 44 55 66 77
@
```

Se o endereço de destino fica dentro do bloco, algumas coisas estranhas acontecem: o bloco entre o endereço inicial e o destino é tratado como sub-blocos e os valores nele contidos são duplicados. Veja:


```

@420.437                                <RETURN>

0420- 00 00 00 00 00 00 00 00
0428- 00 00 00 00 00 00 00 00
0430- 00 00 00 00 00 00 00 00
@420:00 11 22 33 44 55 66 77 88 99 AA BB CC DD
EE FF                                <RETURN>

@426<420.431M                            <RETURN>

@420.437                                <RETURN>

0420- 00 11 22 33 44 55 00 11
0428- 22 33 44 55 00 11 22 33
0430- 44 55 00 11 22 33 44 55
@

```

I.5 - Comparando dois blocos de memória

Pode-se verificar dois blocos de memória, em especial depois de comando M. Caso exista uma discrepância será indicado o endereço onde ela aconteceu e os dois valores discrepantes. O comando VERIFY é do tipo:

```
(destino) < (inicio) . (final)  V
```

por exemplo:

```

@400:00 11 22 33 44 55 66 77  <RETURN>

@410<400.407M                    <RETURN>

@410<400.407V                    <RETURN>

@406:AA                          <RETURN>

@410<400.407V                    <RETURN>

@406-AA (66)
@

```


1.6 Blocos de Memória em Fita Cassette

Há no MD 2 tipos de instruções que permitem armazenar e carregar blocos de memória através de fita cassette. Estes comandos permitem transferir de 1 até 65536 bytes.

Para armazenar há duas versões do comando WRITE (W):

1. (início) . (final) W "nome"
2. (início) . (final) WA

A versão 1, em formato exclusivo, permite armazenar um bloco de memória, associando a ele um nome colocado entre aspas. A versão 2 permite armazenar um bloco de memória no mesmo formato usado pelo computador APPLE - II. Isto permite que blocos de dados possam ser transferidos entre o TK-2000 e o computador APPLE e similares.

A operação com o gravador é similar àquela utilizada com os comandos SAVE e LOAD em linguagem BASIC.

Vejamos um exemplo :

```
@400:0 1 2 3 4 5 6 7 8 9 A B C D E F <RETURN>
```

```
@400.40F <RETURN>
```

```
@400- 00 01 02 03 04 05 06 07
```

```
@408- 08 09 0A 0B 0C 0D 0E 0F
```

```
@400.40FW "TESTE" <RETURN>
```

```
OK
```

```
@
```

Para carregar também há duas versões do comando READ (R):

1. (início) . (final) R "nome"
2. (início) . (final) RA

A versão 1, permite ler um bloco gravado no formato exclusivo do TK - 2000 efetuando a procura de acordo ao nome. A versão 2 permite a leitura de um bloco gravado no formato do computador APPLE.

Agora tentemos carregar o bloco gravado anteriormente:

```
@400:0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 <RETURN>
```

```
@400.40F <RETURN>
```

```
@400- 00 00 00 00 00 00 00 00
```

```
@408- 00 00 00 00 00 00 00 00
```

```
@400.40FR "TESTE" <RETURN>
```

```
TESTE 01 01 WAIT OK
```



```
0400.40F                                <RETURN>
0400- 00 01 02 03 04 05 06 07
0408- 08 09 0A 0B 0C 0D 0E 0F
@
```

Caso o programa tenha sido carregado corretamente aparecerá o símbolo @, caso contrário aparecerá "ERR".

Observe que você pode carregar um bloco numa posição diferente da qual ele foi armazenada.

CAPÍTULO II

CRIANDO E EXECUTANDO PROGRAMAS EM LINGUAGEM DE MÁQUINA

Após ter introduzido um programa em linguagem de máquina para que ele seja executado digite o endereço inicial de execução seguido da letra G e RETURN.

(endereço) G

A letra G é um comando do Modo Monitor que transfere o controle do TK-2000 para o programa no endereço que foi especificado.

O programa em LM, linguagem de máquina, é considerado uma subrotina e para retornar ao MD, você deve colocar no final de seu programa a instrução de retorno (RTS) do 6502. Observe que o programa monitor do TK-2000 possui várias sub-rotinas que você pode utilizar, basta chamá-las através da instrução JRS do 6502.

Como exemplo digite o seguinte programa e execute-o:

```
@400:A9 C1 20 ED FD 18 69 1 C9 DB D0 F6 60 FF
FF FF                                <RETURN>

@400.40F                             <RETURN>

0400- A9 C1 20 ED FD 18 69 01
0408- C9 DB D0 F6 60 FF FF FF
@400G                                <RETURN>
ABCDEFGH IJKLMNOPQRSTUVWXYZ
@
```

Veja o conjunto de instruções do 6502 no Apêndice A

II.1 - Disassembler

Para poder "ler" os mneumônicos Assembly representando um bloco de código binário, o MD do TK-2000 contém um disassembler. O disassembler é um programa que examina um bloco de códigos binários e gera os mneumônicos associados.

Para obter o programa da seção anterior "disassemblado" deve ser indicado o endereço inicial seguido da letra L. Por exemplo, vejamos os comandos envolvidos no programa digitado na seção anterior:

```
@400L                                <RETURN>

0400- A9 C1          LDA  #$C1
0402- A0 ED FD      JSR  $FDED
0405- 18            CLC
0406- 69 01         ADC  #$01
0408- C9 DB         CMP  #$DB
040A- D0 F6         BNE  $0402
040C- 60            RTS
```


040D-	FF	???
040E-	FF	???
040F-	FF	???
0410-	00	BRK
0411-	11 22	ORA (\$22),Y
0413-	33	???
0414-	44	???
0415-	55 66	EOR \$66,X
0417-	77	???
0418-	00	BRK
0419-	00	BRK
041A-	00	BRK
041B-	00	BRK

@

II.2 - Outros Comandos

Pode-se usar a saída de vídeo no modo normal (N) ou no modo inverso (I). Veja como são usados os comandos:

```
@F800.F80F                                <RETURN>

F800- 4C CA EF 4A 08 20 F5 FB
F808- 20 73 FC A5 09 28 90 02
@I                                           <RETURN>

@F800.F80F                                <RETURN>

F800- 4C CA EF 4A 08 20 F5 FB
F808- 20 73 FC A5 09 28 90 02
@N                                           <RETURN>

@F800.F80F                                <RETURN>

F800- 4C CA EF 4A 08 20 F5 FB
F808- 20 73 FC A5 09 28 90 02
@
```

Também pode-se aproveitar o próprio MD para efetuar as operações aritméticas de soma e subtração de dois números hexadecimais:

```
@10+12                                    <RETURN>
=22
@3A-B                                     <RETURN>
=2F
@EE+3                                     <RETURN>
=F1
```

No caso do resultado ser negativo, o MD mostrará o resultado em complemento 2 como se vê abaixo:

@5-8
=FD

<RETURN>

No caso do resultado ser maior que FF, o resultado apresentará apenas os dois algarismos menos significativos.

@FD+5
=02

<RETURN>

Você pode digitar tantos comandos do Monitor quantos quiser numa única linha, contanto que os separe com espaço e o número total de caracteres (incluindo os espaços) seja menor que 254. O único comando que deve ser considerado com cuidado é o comando de alteração (:). Depois dele deve ser inserida uma letra que permita o MD diferenciar o fim deste comando. O comando N normalmente pode ser usado como separador. Efetue o exemplo abaixo:

@300.317 300:00 11 22 33 44 55 66 77 N
300.307 308<B1.C8M 300L <RETURN>

II.3 - Examinando Registradores

O MD reserva 5 posições de memória para os 5 registros internos do 6502. Estes registros são representados pelas seguintes letras:

A = Acumulador
X = Registrador de Índice X
Y = Registrador de Índice Y
P = Registrador de Status do Processador (Status Register)
S = Apontador de Pilha (Stack Pointer)

Os registros acima podem ser consultados a qualquer momento pelo MD através do comando EXAMINE, obtido por um comando CONTROL-E. Siga o exemplo abaixo:

Inicialmente desligue e ligue o computador de modo a "zerar" todas as variáveis do sistema. A seguir digite o comando LM permitindo a passagem para o Modo Monitor e execute a rotina \$FF4A.

>LM <RETURN>
@FF4AG <RETURN>

A seguir entre com o comando EXAMINE, ou seja, CONTROL-E e em seguida pressione a tecla RETURN. O resultado será do tipo indicado abaixo:

A=FF X=00 Y=00 P=B0 S=F4

Os valores à direita do sinal de igualdade correspondem aos últimos valores dos registradores. A rotina \$FF4A (SAVE) atualiza os endereços de memória que são usados para demonstrar os conteúdos dos registradores, portanto sempre que

se queira examinar os conteúdos dos registradores, deve-se antes executar a rotina \$FF4A.

OBS: Os valores apresentados no vídeo no exemplo poderão variar conforme a situação.

Observe que a posição apresentada pelo apontador de pilha "S", deve-se somar \$0100 pois o endereço está na primeira página de memória. Por exemplo:

A=00 X=EA Y=BF P=C0 S=F0

o endereço real da pilha é $\$0100 + \$00F0 = \$01F0$

CAPÍTULO III - O MINI-ASSEMBLER

III.1 - O Mini-Assembler

Nesta seção você aprenderá como preparar, digitar, testar, depurar e executar uma ampla gama de programas em linguagem de máquina.

Um programa fonte, em linguagem Assembly, é escrito usando códigos mnemônicos, etiquetas (labels) e comentários. Este acaba se tornando então em um formato de programação adequado a compreensão humana.

Uma vez que o programador está satisfeito com a versão em linguagem Assembly, ele deve empregar algum mecanismo para traduzir o programa fonte para programa objeto. Um método consiste em buscar numa tabela (Apêndice A) os códigos em hexadecimal correspondentes a cada instrução do programa fonte.

Uma outra alternativa é um programa especial chamado ASSEMBLER que efetua esta tarefa de procura em tabela. O TK-2000 possui o chamado MINI-ASSEMBLER (MA) que, embora não possua todas as poderosas características dos tradicionais Assemblers, elimina o tedioso trabalho e diminui consideravelmente as chances de erro humano na tradução para a linguagem de máquina. Ao efetuar a tradução ele já deposita nos endereços indicados os resultados (Códigos objeto) da tradução.

Para entrar no mini-assembler:

1. Vindo do Modo Monitor/Disassembler:
digite C192G <RETURN>
2. Vindo do BASIC:
digite ASS <RETURN>

O sinal de pronto no mini-assembler (MA) é o ponto de exclamação (!).

Para sair do mini-assembler:

1. Retornando ao Modo Monitor/Disassembler:
digite: \$FF61G <RETURN>
2. Retornando ao BASIC:
digite: \$C2D3G <RETURN> ou RESET

III.2 - Programas com o Mini-Assembler

Suponhamos que você tenha o seguinte programa:


```
LDY #$FF ; INICIALIZA Y
LDX #$FF ; INICIALIZA X
LDA #$00 ; INICIALIZA A
RTS      ; RETORNA A ROTINA PRINCIPAL.
```

Só falta o endereço inicial do programa. Digamos que é \$0320. Responda ao símbolo de exclamação digitando o endereço inicial, dois pontos e a primeira instrução :

```
!0320:LDY #FF
```

Ao apertar RETURN a instrução é assembleada e será substituída por :

```
0320- A0 FF LDY #$FF
!-
```

para a instrução seguinte, você não mais precisa indicar o endereço pois o MA irá atualizá-lo automaticamente. A próxima instrução será digitada sempre deixando um espaço:

```
! LDX #FF
```

Você deve deixar um espaço depois do símbolo de exclamação. Caso contrário aparecerá o indicador de erro (^), e você deverá reescrever a instrução.

Continuando assim você chegará ao final do programa e poderá executá-lo assim como também armazená-lo em fita.

Caso você queira executar um comando do monitor desde o MA, digite inicialmente o símbolo \$.

Exemplo: execute o nosso programa:

```
!$3200
```

Observe que no seu programa deverá sempre existir no final um comando RTS.

Qualquer programa em linguagem de máquina que você criar pode ser chamado no BASIC através da instrução CALL seguida do endereço em notação decimal; quando chamado no MD, deve-se utilizar o endereço em notação hexadecimal seguido do comando G.

Finalmente, digite a título de exemplo este pequeno programa:

```
320: LDY #$0F ; 15 BEEPS.
     JSR $FF3A ; AÇIONE O SOM
     DEY      ; DECREMENTE O CONTADOR
     BNE $0322 ; SE NAO CHEGOU AO FIM, CONTINUE
```


RTS ; RETORNE A ROTINA PRINCIPAL.

Assim sendo, você deverá ter o seguinte programa em linguagem de máquina assembly:

```
0320- A0 F0      LDY #$0F
0322- 20 3A FF    JSR $FF3A
0325- 88          DEY
0326- D0 FA      BNE $0322
0328- 60          RTS
```

Este programa poderá ser executado digitando, como já foi mencionado, \$0320G desde o miniassembler. Se for a partir do MD, com 320G ou se desde o BASIC, com CALL 800.

Ao redigir seu programa, marque todas as etiquetas necessárias, com nomes que dentro do possível tenham relação com a tarefa executada pelo trecho a que ele se referencia.

Na hora de usar o mini-assembler faça uma tabela com os nomes dos labels e na hora de digitar a instrução que o label em questão referencia, escreva na tabela as etiquetas e seus endereços correspondentes, dessa maneira, onde for encontrado uma etiqueta na primeira edição do programa, deve-se colocar o endereço correspondente à etiqueta.

CAPÍTULO IV - O TECLADO

IV.1 Introdução

O teclado é o meio mais comum de entrada de dados do computador. Ele tem por finalidade informar para a CPU a entrada da informação no mesmo instante em que esta é digitada no teclado.

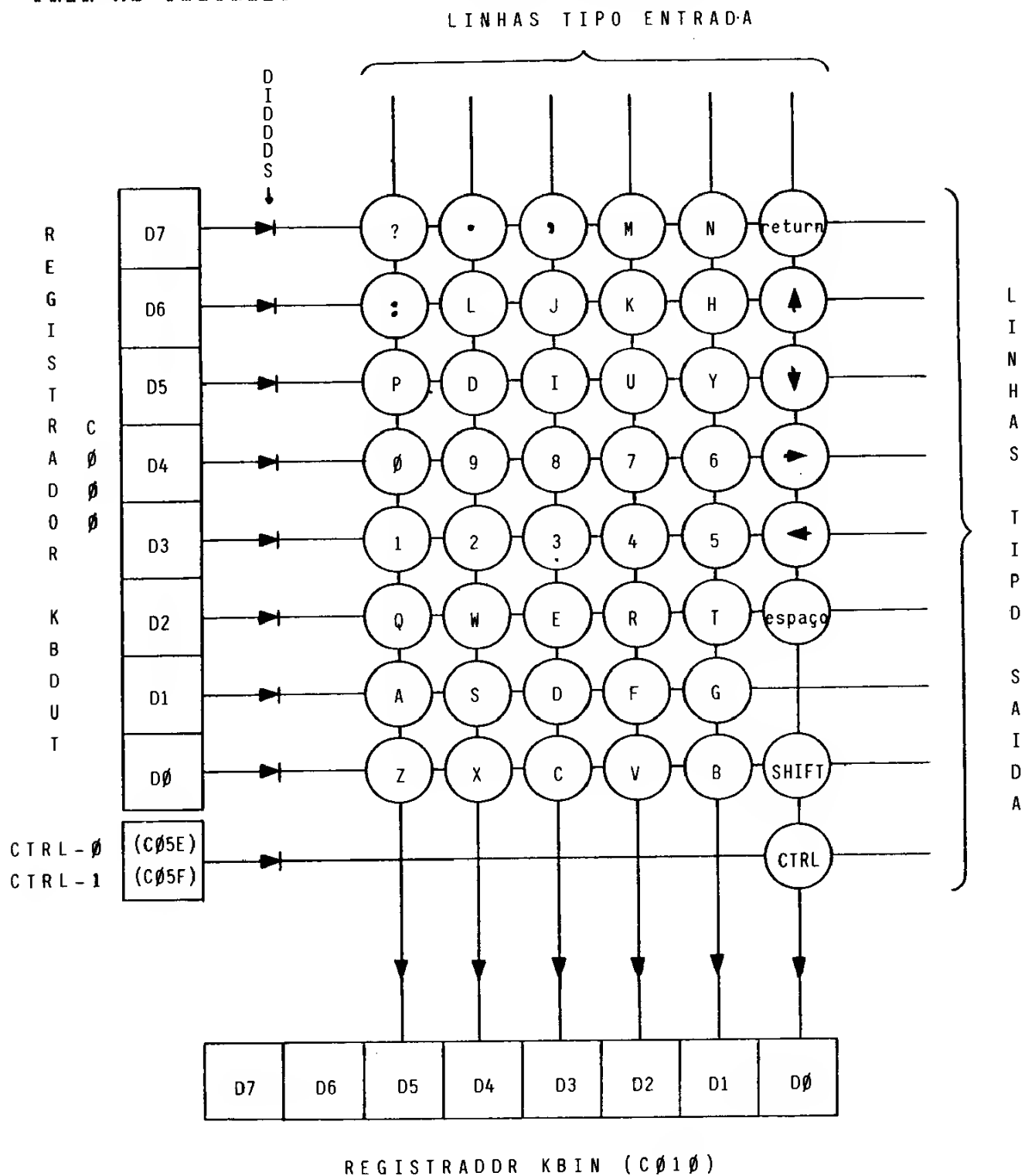


Fig. IV.1

O teclado do TK-2000 é constituído de uma matriz de linhas cruzadas onde cada tecla se encontra num dos cruzamentos. Estas linhas classificam-se em dois tipos: saídas (horizontais) e entradas (verticais). Veja Figura IV.1.

Se uma tecla não estiver pressionada, as duas linhas que se cruzam nela estão separadas, e ao pressioná-la efetua-se um contato físico entre estas duas linhas.

A matriz associa as diferentes teclas do teclado aos bits correspondentes dos registros KBIN e KBOUT.

Basicamente, ao ler KBIN, sem nenhuma tecla pressionada, os bits associados às linhas de entrada indicarão 0. No caso de uma tecla ser pressionada o bit correspondente à linha de entrada adotará o valor lógico existente na outra linha de saída.

Escrevendo no registro KBOUT a informação adequada, podemos identificar através de um método simples, qual tecla tem sido pressionada.

Suponhamos que gostaríamos de identificar se a tecla L foi pressionada para sair de uma rotina.

Uma forma de resolver o problema é acionar a linha de saída de KBOUT com D6=1 e aguardar que a linha de entrada correspondente a tecla indicada por D4 de KBIN, seja 1.

Entre o seguinte programa em linguagem de máquina:

```
0300- A9 40      LDA #40      ; em KBOUT D6=1
0302- 8D 00 C0   STA $C000
0305- A9 10      LDA #10      ; aguarda que D4 de
0307- 2D 10 C0   AND $C010 ; KBIN seja igual
030A- F0 F9      BEQ $0305 ; a 1
030C- 60        RTS
```

Passe ao modo monitor e execute o programa com:

```
0300G          <RETURN>
```

O cursor desaparecerá e só retornará ao vídeo ao pressionar a tecla L. Experimente.

Do mesmo modo, se quisermos identificar a pressão "simultânea" das teclas L e 3, podemos incrementar algumas linhas ao nosso programa:

```
030C- A9 08      LDA #08
030E- 8D 00 C0   STA $C000
0311- A9 08      LDA #08
0313- 2D 10 C0   AND $C010
```



```

0316- F0 E8      BEQ $0300
0318- 60         RTS

```

Execute o programa no endereço \$300 e experimente o teclado.

A linha de saída CTRL é acionada em forma diferente de KBOUT. Para que esta linha tenha nível 0 deve ser acessado o registro CTRL0 (\$C05E), e para obter nível lógico 1 deve ser acessado o registro CTRL1 (\$C05F).

De acordo com o que acabamos de mostrar note que toda a operação de leitura do teclado e identificação foi comandada diretamente pela CPU. Isto provoca notáveis vantagens para o sistema, pois permite identificar não só o pressionamento de teclas individuais como também identifica qualquer combinação de teclas. Devido a esta grande versatilidade, poderá ser aumentado consideravelmente o número de funções geradas diretamente pelo teclado com a mesma mesma quantidade de teclas.

IV.2 - A Rotina de Leitura

O BASIC do TK-2000 possui rotinas cuja função é verificar o teclado. Na memória ROM do TK-2000 existe uma sub-rotina no endereço \$F043 que se destina a receber informações do teclado e identificá-las com os seus respectivos códigos (conforme a tabela 4.1.1)

A seguir é apresentada a tabela com os códigos identificados pela rotina de leitura do TK-2000 em sua versão normal.

TECLA	SOZINHA	CONTROL	SHIFT	CONTROL-SHIFT
-----	-----	-----	-----	-----
Espaço	A0	A0	A0	34
0 *	B0	B0	AA	35
1 !	B1	B1	A1	01
2 "	B2	B2	A2	02
3 #	B3	B3	A3	03
4 \$	B4	B4	A4	04
5 %	B5	B5	A5	05
6 &	B6	B6	A6	06
7 '	B7	B7	A7	07
8 (B8	B8	A8	36
9)	B9	B9	A9	37
: ;	BA	BA	BB	38
, <	AC	AC	BC	39
. > (FIRE)	AE	AE	BE	3A
? /	BF	BF	AF	3B
A	C1	B1	2C	14
B	C2	B2	22	12
C	C3	B3	2A	1A

D		C4	84	28	18
E		C5	85	26	0A
F		C6	86	29	19
G		C7	87	20	10
H		C8	88	21	11
I	-	C9	89	AD	0F
J		CA	8A	1F	1C
K	^	CB	8B	DE	41
L	@	CC	8C	C0	42
M		CD	8D	32	1D
N		CE	8E	23	13
O	=	CF	8F	BD	43
P	+	D0	90	AB	44
Q		D1	91	24	08
R		D2	92	27	0B
S		D3	93	2D	15
T		D4	94	1E	0C
U		D5	95	30	0E
V		D6	96	2B	1B
W		D7	97	25	09
X		D8	98	2F	17
Y		D9	99	31	0D
Z		DA	9A	2E	16
RETURN		8D	8D	8D	40
<-		88	88	88	3C
->		95	95	95	3D
↓		F1	F1	F1	3E
↑		F0	F0	F0	3F
SHIFT		FF	FF	FF	FF
CONTROL		FF	FF	FF	FF

Tabela 4.1.1

O exemplo abaixo irá esclarecer sua utilidade:

Entre o seguinte programa em linguagem de máquina:

```

0300-    20 43 F0    JSR    $FB02
0303-    20 ED FD    JSR    $FDED
0306-    4C 00 03    JMP    $0300

```

A primeira linha chama a rotina de reconhecimento de tecla no endereço \$FB02. A segunda chama a rotina de impressão na tela (vide Apêndice sobre rotinas úteis), e a terceira retorna à primeira linha.

Para rodar o programa acima através do Modo Monitor basta digitar:

```
@300G
```

ou pelo Modo BASIC:

```
>CALL 768
```


Pressione qualquer tecla e observe o vídeo.

Note que:

a) O programa só parará após pressionar as teclas RESET, já que um CONTROL-C será interpretado como um caractere independente.

b) Esta rotina é de apenas 9 bytes e atinge apenas rotinas de ação quase que imediata. Isto a torna excessivamente rápida se comparada ao padrão normal do comportamento do BASIC. Assim, por mais rápido que você seja a sua velocidade ao pressionar qualquer tecla, a rotina irá rodar várias vezes fazendo com que esta tecla se comporte como se fosse auto-repetitiva.

A rotina \$FB02 utiliza a rotina \$F043 que é quem efetivamente efetua a leitura do teclado.

IV.3 Reset

Ao ligar o TK-2000, é efetuado automaticamente RESET na CPU, e ele passa a executar uma rotina que verifica se já recebeu anteriormente RESET ou é esta a primeira vez.

Neste último caso, acontece o chamado "início frio" (cold start), no qual é limpada a tela de vídeo, imprimindo TK2000 na parte superior. Algumas variáveis do sistema são inicializadas, entre elas duas posições de memória que indicaram se é o primeiro RESET. Também é verificada a existência de interface de disco, ou cartucho; caso contrário passa o controle para o BASIC.

Se ele já tinha recebido RESET anteriormente, o ciclo de RESET é chamado de "início em quente" (warm start). Neste caso ele simplesmente retorna ao BASIC.

Existem, como já foi mencionado, 2 posições que permitem à CPU identificar se já aconteceu um RESET anterior. A idéia por trás deste processo é que é praticamente impossível duas posições manterem uma relação ou algoritmo constante entre si. O início em frio é considerado quando eles não mantêm esta relação.

As posições \$3F2 e \$3F3 contêm o vetor de retorno do RESET em "início em quente". Normalmente o endereço é \$C2D3. O byte \$3F3 contém portanto #C2. A relação aplicada é efetuar a função EXCLUSIVE OR do conteúdo de \$3F3 com o valor #A5, e guardar o resultado no byte seguinte, ou seja, \$3F4.

Vejamos como é feito o cálculo:

CAPÍTULO V - ORGANIZAÇÃO DA MEMÓRIA

V.1 Descrição das páginas de memória

O microprocessador 6502 pode acessar diretamente com um total de 65536 posições de memória distintas. Pode ser feita uma analogia desta memória com um livro de 256 "páginas", com 256 posições de memória cada página. Por exemplo "página \$30" seria constituída por 256 posições de memória a partir da posição \$3000 até a posição \$30FF. O endereço de qualquer posição de memória requer 2 bytes, e isto permite relacionar o byte mais significativo como sendo o número da página e o menos significativo como sendo a localização dentro da página.

As 256 páginas de memória do TK-2000 compreendem funcionalmente três categorias: RAM, ROM e área de I/O (entrada e saída). O mapa básico de memória do TK-2000 está na figura abaixo (Vide também Apêndice F).

MAPA DE MEMÓRIA DO SISTEMA

PÁG. NUM:

Decimal Hexa

0	\$00	
1	\$01	
2	\$02	
"	"	
"	"	RAM (192 páginas)
"	"	
190	\$BE	
191	\$BF	
192	\$C0	I/O (1 Página)
193	\$C1	
"	"	
"	"	ROM (63 Páginas)
"	"	
254	\$FE	
255	\$FF	

Tabela 6.1

V.2 - Memória RAM

A área utilizada pela memória RAM é compreendida entre a página 0 e a página 191 (ou \$BF em hexa).

O uso da RAM é diferenciado pela função que foi atribuída a cada área:

- Página Zero:** Devido à arquitetura do 6502, esta página tem características especiais e é usada para armazenar as variáveis do sistema.
- Página Um :** O 6502 reserva esta página como seu "stack" (pilha de dados). Mesmo que o TK-2000 geralmente utilize menos da metade desta página para esta função, é muito difícil determinar o limite de utilização em qualquer instante. Assim sendo, não é recomendável armazenar dados nesta área.
- Página Dois:** A sub-rotina GETLN, utilizada como entrada da linha digitada pelo sistema, usa a página 2 como buffer de entrada.
- Página Três:** Esta área está reservada parcialmente para os periféricos e vetores de endereço do sistema. Nesta página, você pode utilizar os endereços desde \$300 até \$3EF para seus próprios programas em linguagem de máquina. Também nos três bytes de \$3F5 até \$3F7 você pode inserir uma instrução de JMP para outro endereço que contenha um programa em linguagem de máquina e que será executado em BASIC pela instrução "&" (Tecla SHIFT-6). Idem para os três bytes de \$3F8 até \$3FA, sendo que estes serão executados pela instrução "TK2000" (Tecla CONTROL-7).
- Páginas Quatro a Sete:** Esta área é reservada para o programa MONITOR, algumas variáveis do sistema e para os periféricos.
- Páginas \$8 a \$1F:** Disponível para o usuário. Os programas em BASIC começam normalmente na página 8. Também pode ser usado para programas em linguagem de máquina.
- Páginas \$20 a \$3F:** Área reservada para a primeira página de vídeo para texto, baixa e alta resolução.
- Páginas \$40 a \$9F:** Disponível para o usuário.
- Páginas \$A0 a \$BF:** Área reservada para a segunda página de vídeo para texto, baixa e alta resolução.
- V.3 Memória ROM**
- A área de ROM efetivamente disponível para o 6502 ocupa as páginas \$C1 até \$FF.
- A ROM contém uma sequência de códigos a serem interpretados pelo 6502 que permitem a operação do TK-2000 como computador. O BASIC, o monitor-dissassembler e o mini-assembler estão armazenados na ROM.
- Algumas rotinas específicas e de utilidade para o usuário estão relacionadas no APÊNDICE D deste manual.

V.4 I/O

Entradas e saídas (I/O) são registros especiais que permitem à CPU 6502 controlar tanto periféricos externos quanto as "chaves" internas que controlam o comportamento do TK-2000 COLOR. A tabela abaixo apresenta estes endereços:

FUNÇÃO	REGISTRO	ENDEREÇO	COMENTARIO
Saída de dados	KBOUT	\$C000	
Entrada de dados	KBIN	\$C010	
Saída de gravador	CASOUT	\$C020	
Som	SND	\$C030	
Video colorido	VCL	\$C050	
Video branco/preto	VBP	\$C051	
MOTOR A	MTA0FF	\$C052	Desligado
	MTA0N	\$C053	Ligado
Página de video	VD1	\$C054	Primeira
	VD2	\$C055	Segunda
MOTOR B	MTB0FF	\$C056	Desligado
	MTB0N	\$C057	Ligado
Strobe de impress.	STR0	\$C058	Nível Baixo
	STR1	\$C059	Nível Alto
Selecionador de RAM/ROM	RO	\$C05A	ROM
	RA	\$C05B	RAM
Tecla CONTROL	CTRL0	\$C05E	Baixo
	CTRL1	\$C05F	Alto

V.5 - Video

Como foi visto no item V.1, o TK-2000 reserva duas áreas de memória para as duas páginas de video disponíveis.

Será visto no capítulo VII que cada uma destas áreas de memória será utilizada nas três categorias de video: texto, baixa e alta resolução.

V.6 - TK-2000 II Versões 64K e 128K de RAM

Se você possui o TK-2000 II 64K, consulte o APENDICE G, incluído no final deste manual.

Se o seu TK-2000 II se apresenta em versão de 128K, veja o APENDICE H, também incluído na parte final deste livro.

PAGINA FOI DEIXADA EM BRANCO PROPOSITADAMENTE

VI.1 - Introdução

No Manual de Operação do TK-2000, Capítulo XV, é apresentado o comando SOUND gera uma nota musical com tom e duração programável; também é apresentado alguns efeitos especiais gerados diretamente em linguagem de máquina.

Este capítulo irá complementar os conceitos já expostos apresentando o modo de operação do canal de som em forma direta através de programas em linguagem de máquina.

VI.2 - Canal de Som

Antes de iniciar qualquer programação ou exemplo, é importante que o "coração" de todos os efeitos sonoros seja apresentado. Esta principal ferramenta é o registro SND localizada no endereço \$C030 da página de I/O.

Através da programação já existente no TK-2000, qualquer acesso ao endereço \$C030 (ou 49200 decimal) gerará uma alteração de nível lógico (0 para 1 ou vice-versa) no canal de som da TV provocando um click no alto-falante.

Este acesso ao endereço \$C030 pode ser dado através de um comando em linguagem de máquina (que será visto adiante) ou através do BASIC usando um simples comando, por exemplo:

```
>PRINT PEEK(49200)
```

Para sentir melhor o efeito do comando acima digite o seguinte programa:

```
>10 P = PEEK(49200)
>20 RUN
```

após rodar o programa acima será possível sentir o resultado de uma série de click's no alto-falante de seu TV.

No exemplo acima, notamos que a frequência do som gerado era relativamente baixa (cerca de 70 Hz). Isto ocorreu devido a fato de o endereço \$C030 ter sido acessado através de um programa BASIC que, como sabemos é mais lento que programas escritos em linguagem de máquina.

A maior versatilidade se obtém acessando diretamente em linguagem de máquina o registro SND. O controle de tonalidade seria então produzido por intervalos regulares programados de modo a reproduzir sons que possam se aproximar de notas musicais.

VII.3 - Programando Tonalidades musicais

Abaixo é apresentada uma rotina em linguagem de máquina muito utilizada na formação de tons.

```
0302-    AD 30 C0    LDA    $C030
0305-    88        DEY
0306-    D0 05      BNE    $030D
0308-    CE 01 03   DEC    $0301
030B-    F0 09      BEQ    $0316
030D-    CA        DEX
030E-    D0 F5      BNE    $0305
0310-    AE 00 03   LDX    $0300
0313-    4C 02 03   JMP    $0302
0316-    60        RTS
```

Programa I.

No exemplo acima, o endereço \$0300 deve conter um número (entre 0 até 255, claro!) que controla o tom da nota produzida, enquanto que o endereço \$0301 contém um byte que indicará a duração da nota emitida.

Para poder sentir, inicialmente, uma pequena demonstração da rotina apresentada, bastará digitar:

03020

ou no modo BASIC simplesmente:

>CALL 770

O princípio de trabalho desta rotina consiste simplesmente em um entrelaçamento de loops. Inicialmente no primeiro elo de tempo decrementa-se o registrador Y (cujo valor inicial não é determinado pela rotina) e quando Y é zero, irá decrementar o número guardado no endereço \$0301 até chegar ao zero finalizando assim a rotina. Assim, percebemos que o valor contido em \$0301 controla a duração da nota, e que a rotina irá passar pela instrução DEY aproximadamente $256 * (T-1)$ vezes, onde T representa o conteúdo da posição \$0301. A exceção para isso é que se $T = 0$ o resultado é assumido como sendo 256.

O registrador X é carregado com o conteúdo da posição \$0300 (que nós chamaremos de P) e é decrementado a cada passo completo do loop. Quando X torna-se zero, o endereço \$C030 é acessado (gerando outro click no alto-falante) e o registro X é "refrescado" com P, ou seja, o registro que ficou igual a zero irá ser novamente carregado com o valor de P. Consequentemente é evidente que P controla o espaço de tempo entre cada chamada a \$C030, consequentemente controlando o tom da nota. Diminuindo o valor de P aumenta-se o tom da nota gerada com a importante exceção que

P=0 produz a nota mais grave possível produzida por esta rotina.

Esta subrotina pode ser acessada através de comandos em linguagem de máquina usando as seguintes linhas (nos exemplos T=0 e S = 0)

```
LDA #$00
STA $0300
LDA #$00
STA $0301
JSR $0302
```

"
"
"

ou alternativamente através do modo BASIC:

```
T = 0
P = 0
POKE 768,P
POKE 769,T
CALL 770
```

"
"
"

A simples título de experiência, após introduzir o programa em linguagem de máquina no endereço correto, digite as seguintes linhas em BASIC:

```
>10 T = 120
>20 P = 96
>30 POKE 768,T
>40 POKE 769,P
>50 SOUND P,T
>60 CALL 770
```

O programa acima gerará duas notas de mesma duração e de mesmo tom. Assim conclue-se que, no caso da subrotina apresentada, que os valores apresentados nas tabelas XV.1 e XV.2 no Manual de Instrução são também válidos para formação de partituras musicais do mesmo modo que o comando SOUND. Somente que, neste caso, um maior número de linhas de programa será requisitado para executar a mesma tarefa que o comando SOUND.

Capítulo VII - Vídeo

VII.1 Modo Texto

As mais importantes partes de seu sistema TK-2000 encontram-se no teclado e na saída de vídeo, pois são o primeiro elo de ligação entre o usuário e o interior do sistema.

Este capítulo irá discutir os formatos de entrada e saída de informações através do teclado e do vídeo.

Formato Standard de Texto: A tela de vídeo do TK-2000 está arranjada de modo a exibir um máximo de 40 caracteres em cada uma das 24 linhas de texto. Isto permite que 960 caracteres possam ser exibidos simultaneamente na tela.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39		
0																																									
1																																									
2																																									
3																																									
4																																									
5																																									
6																																									
7																																									
8																																									
9																																									
10																																									
11																																									
12																																									
13																																									
14																																									
15																																									
16																																									
17																																									
18																																									
19																																									
20																																									
21																																									
22																																									
23																																									

Figura 7.1. Localizações dos caracteres na tela

A figura 7.1 mostra a tela de vídeo dividida em 960 possíveis localizações de caracteres. A numeração que acompanha indica apenas o endereçamento dado a cada linha ou coluna.

As colunas são numeradas de 0 a 39 formando 40 colunas e as linhas são numeradas de 0 a 23 perfazendo 24. Assim, a primeira posição no canto superior esquerdo da tela é descrita como coluna 0, linha 0. O caracter localizado no canto inferior direito da tela será localizado como linha 23, coluna 39. O caracter localizado próximo ao centro da tela será localizado como coluna 19, linha 11.

Virtualmente todas as operações envolvendo o vídeo usam este formato de numeração, coluna e linha, de um modo ou de outro. As operações normais de impressão no vídeo ocorrem da esquerda para a direita. É possível, no entanto, alterar a ordem normal de impressão de eventos e imprimir uma sequência de caracteres seguindo uma outra direção que você mesmo tenha definido.

VII.1.1 Funcionamento do Cursor

Na maior parte das operações, o cursor sempre indica onde o próximo caracter será impresso na tela. O cursor é representado por um retângulo cheio no BASIC, no modo monitor e no mini-assembler. O sistema interno de manipulação do cursor usa dois endereços específicos de memória que contêm as coordenadas coluna e linha do cursor.

Supondo que esteja no modo BASIC, digitando arbitrariamente uma sequência de letras e números, você verá o símbolo do cursor avançando ao longo da linha, sempre indicando exatamente onde o próximo caracter será apresentado. Se você preencher a linha e continuar digitando mais caracteres, o cursor automaticamente passará para a próxima linha voltando para a primeira coluna e assim sucessivamente.

O mesmo tipo de comportamento ocorre quando é impresso um texto na tela. A única diferença é que o símbolo do cursor não aparece na tela. Isto acelera o trabalho do computador evitando a perda de tempo em apresentar o cursor quando um comando de impressão é executado. Em outras palavras, o cursor só indicará a próxima posição do caracter a ser projetado quando o computador estiver aguardando alguma entrada de informação pelo teclado.

Exemplificando, ao executar o comando PRINT "HELLO", o computador imprime o caracter H em uma dada localização coluna e linha, avança um endereço de coluna, imprime o E, avança mais um endereço de coluna, depois o L e assim por diante.

VII.1.2 Controlando a posição do cursor através do Comando PRINT

O comando PRINT faz com que o computador "imprima" um string alfanumérico ou valores numéricos na tela do vídeo. E, a menos que o computador esteja programado para executar de outro

modo, a informação é "impressa" um caracter por vez, de esquerda para direita, com o "invisível" cursor "indicando o caminho".

Todos as opções de controle do cursor através de PRINT estão descritas no Manual de Operação do TK-2000 página 41; enquanto que os comandos TAB, HTAB e VTAB estão descritos na página 90 do mesmo manual.

VII.1.3 Os registradores de posição do cursor

Existe na memória do TK-2000 duas posições de memória que indicam exatamente onde o cursor está localizado num dado momento e sob qualquer modo de operação. Os valores numéricos contidos nestas localizações seguem o mesmo formato de endereçamento coluna e linha descrito neste capítulo.

Estes dois endereços de memória serão denominados por CH e CV:

CH endereço \$24 da RAM - este registrador especifica a coluna em que se localiza o cursor (0-39).

CV endereço \$25 da RAM - este registrador especifica a linha em que se localiza o cursor (0-23).

e através de comandos PEEK e POKE é possível ter o controle total sobre o posicionamento do cursor como será visto adiante.

O endereço de memória 36 (\$24) sempre contém o endereço do cursor horizontal na tela; assim incluindo um comando BASIC como:

```
PRINT PEEK(36)
```

o seu programa indicará o valor de CH para você. Mas se você tentar este comando no modo imediato, o sistema sempre responderá respondendo 0. Isto ocorrerá porque um comando no modo imediato sempre é executado somente depois da tecla RETURN ser pressionada. A tecla RETURN sempre provocará uma mudança do cursor para a linha seguinte (linefeed) levando o cursor para a coluna endereço 0. Consequentemente um comando PRINT no modo imediato é virtualmente inútil.

Aplicando o comando PRINT PEEK(36) dentro de um programa, será possível visualizar o desempenho do registrador de endereço de coluna. Tente esta pequena demonstração:

```
>10 FOR I = 1 TO 4  
>20 PRINT "*";  
>30 NEXT I  
>40 PRINT PEEK(36)
```


rodando o programa acima aparecerá na tela a seguinte resposta:

****4

O comando FOR-NEXT imprimiu asteriscos nas colunas 0, 1, 2 e 3. A seguir o sistema avançou o cursor para a próxima coluna, e então o comando da linha 40 imprimiu a atual posição do cursor que, no caso, era 4.

A seguir é apresentado um outro programa que utilizará novamente o comando PRINT PEEK(36):

```
>10 INPUT "ENTRE COM O ENDEREÇO DA COLUNA (0-39)";COL
>20 FOR N = 0 TO COL : PRINT "X"; : NEXT N
>30 PCL = PEEK(36)
>40 PRINT : PRINT PCL
>50 GOTO 10
```

Inicialmente o programa questiona qual o endereço de coluna que se deseja visualizar na linha seguinte de texto. A seguir o programa imprimirá uma série de "X's" até o endereço requerido na linha 10. O próximo passo será armazenar a posição final do cursor na variável PCL que será então impressa na linha seguinte.

Tente, por exemplo, omitir o ponto-e-vírgula na linha 20 e veja o que ocorrerá.

Usando a mesma linha de raciocínio, será simples assegurar por si mesmo que CV (endereço 37 da RAM) sempre indica o endereço da linha no qual o cursor se localiza. Para isso, basta requerer, a qualquer instante, o valor de CV através do comando:

```
PRINT PEEK(37)
```

fazendo isto, a tela mostrará um número entre 0 a 23, dependendo do valor atual do cursor.

Abaixo é apresentado um programa mostrando a utilização do registro CV:

```
>10 INPUT "ENTRE COM UM ENDEREÇO DE LINHA (0-23)";
    LIN
>20 HOME
>30 FOR N = 0 TO LIN : PRINT "X" : NEXT N
>40 PLN = PEEK(37)
>50 PRINT PLN
>60 GOTO 10
```

Inicialmente o programa irá requisitar um valor de linha. A seguir limpará a tela e imprimirá o caracter "X" no mesmo número de linhas requisitado pela linha 10. A linha 40 então armazenará o valor do endereço da linha seguinte que será exibido através da linha 50.

VII.1.4 Formatos alternativos de texto

O TK-2000 COLOR oferece algumas alternativas para o formato de texto, além do descrito no item anterior. As alternativas incluem os caracteres inversos e o tamanho da tela para o texto. É certamente possível rodar algumas úteis e sofisticadas rotinas de texto sem utilizar estas alternativas, mas elas sem dúvida oferecem grande capacidade de sofisticar e valorizar qualquer programa.

Quanto a forma dos caracteres, se em inverso ou não, todas informações sobre este assunto já foram abordadas no Manual de Operação no item VII.4.7.

O texto normal do TK-2000, é formado por 24 linhas tendo 40 caracteres cada. Isto representa um máximo de 960 caracteres a serem apresentados simultaneamente conforme mostra a figura 7.1. Ao tentar a impressão de mais de 960 caracteres no mesmo instante, ocorrerá o "scrolling", ou seja, a tela inteira se moverá para cima fazendo com que a primeira linha desapareça surgindo um novo texto na linha 23 (considerando a linha de cima como linha 0).

Sempre que se ligar o sistema, ou se aplicar o comando RESET, automaticamente o computador irá para seu modo normal de texto (24 linhas x 40 colunas, em modo normal). Utilizando-se uma série de comandos será possível comandar todas as dimensões da tela de vídeo, formando uma espécie de "janela" dentro do próprio monitor de TV. Estes comandos são aplicados utilizando-se de comandos POKE's com valores apropriados à família de 4 endereços especiais da RAM.

VII.1.5 Regulando a primeira coluna de texto

Recordando a figura 7.1 vemos que ela apresentou cada linha organizada em 40 diferentes localizações de colunas que são numeradas de 0 a 39. A coluna endereçada por 0 representa a localização do caracter coluna da extrema esquerda da tela, e o endereço 39 representa a localização da caracter localizado na extrema direita.

Através do conteúdo da posição 32 (\$20) da memória RAM do TK-2000 é possível determinar o endereço da coluna para o primeiro caracter a ser impresso em cada linha.

Normalmente, o valor contido na posição 32 é zero. Isto significa que cada linha de texto começará da coluna zero do vídeo, na extrema esquerda da tela. Assim, ao digitar:

```
>PRINT PEEK(32)
```

o resultado será:

0

Entrando a seguinte instrução:

>POKE 32,15

o símbolo de pronto (>) passará a ocupar uma posição próxima ao meio da linha. Pressionando-se várias vezes a tecla RETURN poderá se confirmar que cada linha de texto está começando na coluna de endereço 15.

Entrando agora:

>POKE 32,0

retornará o início da linha ao valor original .

Através do comando POKE é possível aplicar à posição 32 outros valores inteiros que estejam entre 0 e 39. Porém trabalhando com um valor diferente de 0 não se deve transbordar a linha de texto, pois isto pode causar erros além de colocar dados em localizações impróprias.

A seção seguinte irá descrever como terminar com os problemas inerentes a transbordamentos de linha quando a posição 32 for diferente de zero.

VII.1.6 Regulando o número de caracteres por linha

O endereço 33 (\$21) da memória RAM do TK-2000 irá determinar até que coluna os caracteres podem ser impressos em cada linha.

Quando o TK-2000 está em seu modo normal, seja quando recém ligado ou quando as teclas RESET são pressionadas, o valor contido na posição 33 será 40, indicando a possibilidade de impressão até a coluna 40.

A fim de perceber a utilidade desta posição de memória, siga a sequência abaixo:

1. Pressione as teclas RESET. Isto irá garantir o tamanho normal da janela.
2. Digite:

PRINT PEEK(32), PEEK(33)

Desde que o sistema está agora com as dimensões normais da janela, os valores lidos pela instrução acima serão 0 e 40, respectivamente. Isto significa que o texto começa na coluna endereço 0 e que, em cada linha avança até a coluna 40.

3. Entre com:

POKE 33,10

Esta operação irá definir a coluna direita da janela de vídeo como 10. Digite um string longo de caracteres arbitrários até ultrapassar o limite de dez e note o resultado.

4. Digite:

PRINT PEEK(32), PEEK(33)

O resultado da instrução acima será 0 para o início da linha e 10 para o comprimento máxima desta.

5. Faça:

POKE 33,40

Isto deverá retornar o sistema para o modo normal de 40 caracteres por linha. Confirme isso digitando um string longo e repetindo novamente o item 4.

6. Agora faça a seguinte experiência:

```
>NEW
>10 POKE 32,10 : POKE 33,30
>20 PRINT : PRINT PEEK(32),PEEK(33)
>RUN
```

você verá:

10 30

OBS: O valor da posição 33 nunca deve ser menor que o da posição 32.

A seguir serão tratados os endereços de controle do endereços de linhas..

VII.1.7 Regulando a posição da linha de topo

O endereço da linha de topo da janela de texto está registrada na posição 34 (\$22) da memória RAM do TK-2000. Normalmente o sistema contém nesta posição o valor 0, indicando que a primeira linha de texto deve aparecer no topo da tela.

Para se poder compreender o significado da posição 34 basta digitar:

```
>POKE 34,10
>HOME
```

verificando que agora o novo topo de tela passou a ser a linha dez da tela.

Repare que o comando HOME foi necessário, pois sem ele

não seria possível localizar o novo topo da tela.

Ao se trabalhar com baixa ou alta resolução, uma das mudanças que ocorrem na passagem do modo texto para o sistema de resolução é que o valor de PEEK(34) passa automaticamente a 20 qualquer que seja o valor (inclusive 0) armazenado anteriormente.

No caso do controle de topo de tela, além do comando RESET, o comando TEXT também pode desativar qualquer alteração imposta a ele fazendo com que a posição 34 retorne ao valor 0.

Os valores permitidos para se modificar o topo de tela deverão estar na faixa entre 0 e 22.

VII.1.8 Regulando a última linha da tela

A posição de memória 35 (\$23) da RAM do TK-2000 contém o número da última linha do texto. Portanto no modo normal, o valor contido nessa posição é 24.

A faixa útil deste comando é de 1 a 24. Note que o valor contido nesta posição não obedece exatamente a mesma notação de endereçamento que começa a contagem desde 0. Assim sendo bastará acrescentar 1 ao endereço convencionado e aplicá-lo à posição 35 da memória através do comando POKE.

VII.1.9 Programando a janela de texto

A tabela 7.1 apresenta um sumário de todos os endereços da RAM que controlam a janela do modo texto do TK-2000.

SIGNIFICADO DA INSTRUÇÃO	ENDEREÇO NA RAM	FAIXA DE VALORES	VALOR NORMAL
Endereço da co- luna do primeiro caracter de cada linha.	32	0-39	0
Endereço da co- luna do último caracter de cada linha.	33	1-40	40
Endereço da li- nha de topo.	34	0-22	0
Endereço da úl- tima linha da janela de vídeo.	35	1-24	24

Tabela 7.1

VII.2 Modo de operação em baixa resolução

Uma das mais interessantes características do sistema TK-2000 é o versátil e colorido modo gráfico de baixa resolução. Este modo, quando usado apropriadamente, poderá proporcionar interessantes e práticos gráficos coloridos.

A área ocupada pelo modo de baixa resolução na memória RAM é a mesma ocupada pelo modo texto.

O modo normal de baixa resolução gráfica permite a mistura de gráficos e texto simultaneamente. A área de texto ocupa as quatro últimas linhas (linhas 20 até 23), e a de gráficos que ocupam o restante da tela.

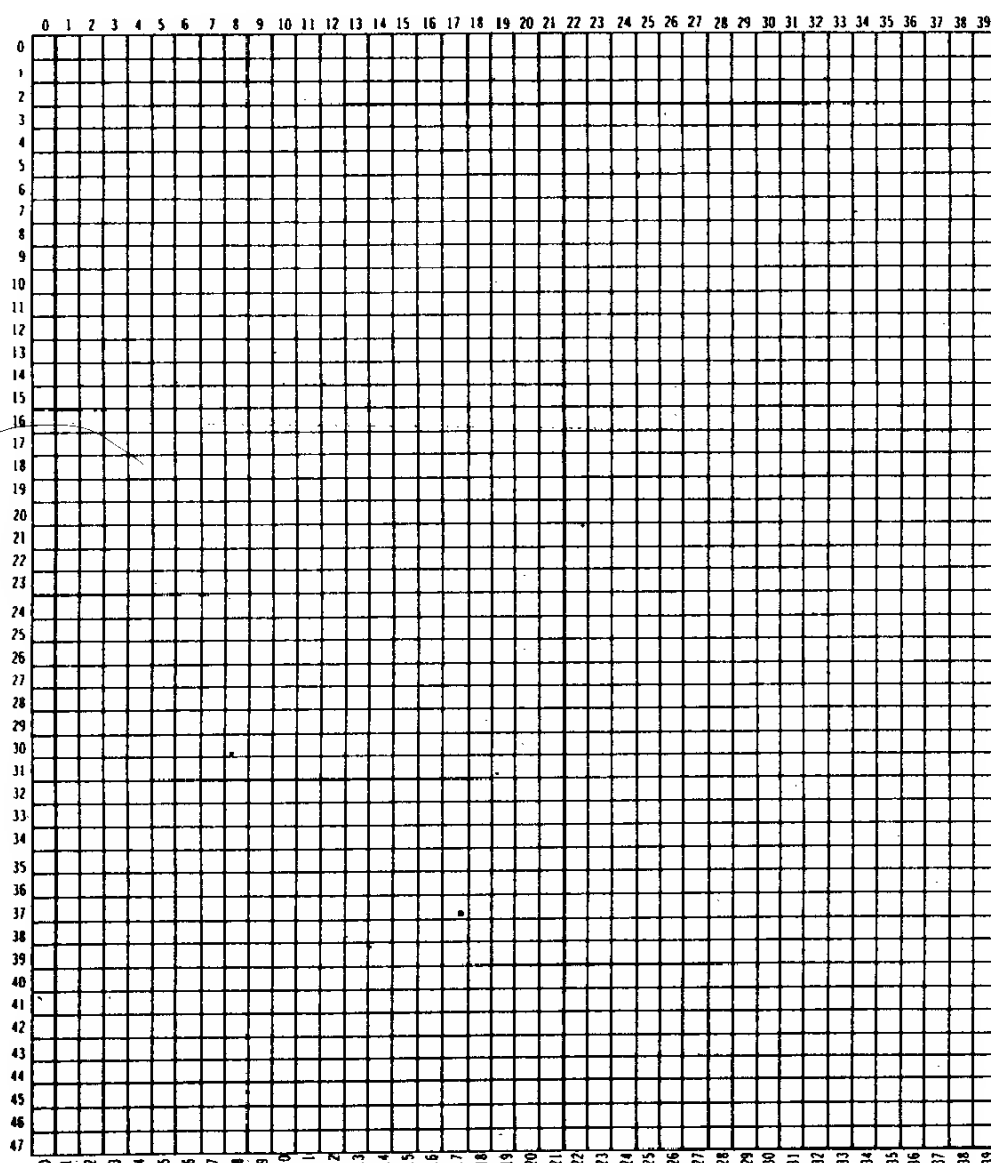


Fig. 7.2

A área gráfica está dividida em 40 linhas de 40 localizações gráficas. Assim a tela proporcionará 1600 posições (ou localizações) na tela onde se imprimirá pequenos retângulos coloridos. As 40 colunas de cada linha estão endereçadas exatamente da mesma forma que as localizações de caracteres de texto - 0 até 39. Da mesma forma, as 40 linhas de baixa resolução podem ser intituladas de 0 até 39. (Vide fig. 7.2)

Será simples apreciar o fato que há 40 linhas de posições gráficas ocupando o espaço que é normalmente ocupado por 20 linhas de texto. Em outras palavras, há duas vezes mais linhas gráficas por área de vídeo que com texto.

Observe que as linhas 40 a 47 correspondem, na realidade, às 4 linhas de texto.

As instruções relativas ao modo gráfico de baixa resolução constam do Manual de Operação (Capítulo X).

É importante perceber que o mapa apresentado acima será muito útil para "rascunhar" o desenho antes de iniciar qualquer programação. Assim ficará muito mais fácil obter, através dos comandos PLOT, HLIN ou VLIN, resultados bonitos em pouco tempo.

VII.3 Operações no modo de alta resolução

VII.3.1 Introdução

O modo de alta resolução é uma base fértil para interessantes experiências e trabalhos sofisticados com o seu TK-2000.

Este modo permite ter controle de cada ponto da tela; sendo que esta é dimensionada com 280 pontos horizontais por 192 pontos verticais. A aparência da imagem do vídeo é grandemente afetada pelo grau de resolução da figura. Ao verificar a formação de baixa resolução foi notado que a construção de uma figura era feita através de elementos gráficos com dimensão metade de um caracter normal. Em alta resolução, o controle de pontos individuais permite criar figuras com nitidez muito superior.

Nas seções seguintes serão apresentados uma série de métodos de uso da alta resolução. Serão abordados principalmente, o mapeamento da memória, e o controle dos pontos gráficos individualmente e em conjunto, na formação de figuras.

VII.3.2 Memória ocupada pela alta-resolução

A área ocupada pela alta resolução é a mesma que a ocupada pelo modo texto e pela baixa resolução. A diferença, convém notar, reside na maneira com que os pontos da tela são controlados. A primeira página, página 1, está localizada deste 8192 até 16383 (\$2000-\$3FFF), e a segunda página, página 2, reside desde a posição 40960 até 49151 (\$A000-\$BFFF). A maioria dos exemplos aqui apresentados utilizarão a página 1, mas as técnicas e programas discutidos trabalharão identicamente na página 2 simplesmente adicionando 32768 (\$8000) aos endereços.

O primeiro exemplo apresenta um programa designado para apresentar todos os pontos de alta resolução sequencialmente. Partindo da posição 8192 liga o primeiro bit do primeiro byte, apaga-o e liga o segundo; ao terminar o primeiro byte começará com o segundo (8193) e assim por diante até completar toda a tela.

Ao aplicar um comando POKE, o byte inteiro será aplicado na posição de memória. Assim, teremos que aplicar 8 comandos POKE com 8 valores diferentes de modo a poder aplicar um valor 1 a cada bit individualmente. A seguinte sequência em programa BASIC gerará os oito valores requeridos.

```
FOR J = 1 TO 8
  X = 2 ^ (8 - J)
NEXT
```

Quando J é igual a 1, X é $2^7 = 128$, J igual a 2 dá X igual a 2^6 , e assim por diante como indica a figura 6.2

J	$2^{(8-J)}$	BINARIO		HEXADECIMAL
1	2^7	1000	0000	\$80
2	2^6	0100	0000	\$40
3	2^5	0010	0000	\$20
4	2^4	0001	0000	\$10
5	2^3	0000	1000	\$08
6	2^2	0000	0100	\$04
7	2^1	0000	0010	\$02
8	2^0	0000	0001	\$01

Tabela 7.2 - Potência de dois

Como foi dito, cada valor terá somente um bit em 1, e note também que, o índice "J" foi de 1 para 8 enquanto que os bits foram acendendo da esquerda para a direita. O programa a seguir usa este algoritmo sucessivamente aplicando um comando POKE para cada situação na memória de alta resolução.

```

>10 HGR : REM ACERTA O MODO AL-RES
>20 X = PEEK(49234): REM UTILIZA TODA PAGINA
>30 FOR I = 8192 TO 16383
>40 FOR J = 1 TO 8
>50 X = 2 ^ (8-J)
>60 POKE I,X : REM ENTRA COM O PONTO
>70 FOR A = 1 TO 2: REM REALIZA UM ATRASO
>80 NEXT A
>90 NEXT J
>100 NEXT I
>110 CALL 65338 : REM PROVOCA UM BEEP
>120 END

```

Ao observar o resultado causado pelo programa, verifica-se que em cada byte a sequência de acendimento foi da direita para a esquerda e que, na verdade, apenas 7 bits acendiam e não 8 em cada byte.

Este fato deve-se a que na tela é projetada a imagem do conteúdo da memória. Se o bit mais a direita está em 1 então o ponto mais a esquerda "acende". A razão de que apenas 7 pontos apareçam para cada posição de memória é que o bit mais significativo de cada byte de alta resolução é usado para controlar as cores. Veja a tabela 7.3:

10000000	!	NAO	NAO	NAO	NAO	NAO	NAO	NAO
01000000	!	NAO	NAO	NAO	NAO	NAO	NAO	SIM
00100000	!	NAO	NAO	NAO	NAO	NAO	SIM	NAO
00010000	!	NAO	NAO	NAO	NAO	SIM	NAO	NAO
00001000	!	NAO	NAO	NAO	SIM	NAO	NAO	NAO
00000100	!	NAO	NAO	SIM	NAO	NAO	NAO	NAO
00000010	!	NAO	SIM	NAO	NAO	NAO	NAO	NAO
00000001	!	SIM	NAO	NAO	NAO	NAO	NAO	NAO

Tabela 7.3

A seguir é apresentado o mapa da memória da alta resolução.

Pelo exemplo, é possível observar que endereços consecutivos na memória não representam diretamente posições consecutivas na tela de vídeo. A relação entre o endereço na memória e a posição será esclarecida adiante, e está relacionada no Apêndice C.

Para ativar um ponto é necessário encontrar inicialmente o endereço da posição, a linha de pontos que contém a posição desejada e finalmente o ponto que se quer ativar.

VII.3.3 Formação de figuras em alta resolução

Através dos dois exemplos seguintes será possível entender a formação de figuras em alta resolução. Os métodos aqui descritos utilizarão como forma de acesso à tela de AR através de comandos POKE. Existem, ainda, outras formas de acesso ao modo de AR que estão descritos no Capítulo X do Manual de Operação do TK-2000.

EXEMPLO :

Supondo que desejemos escrever na tela o vigésimo ponto da esquerda para a direita, na décima linha de pontos.

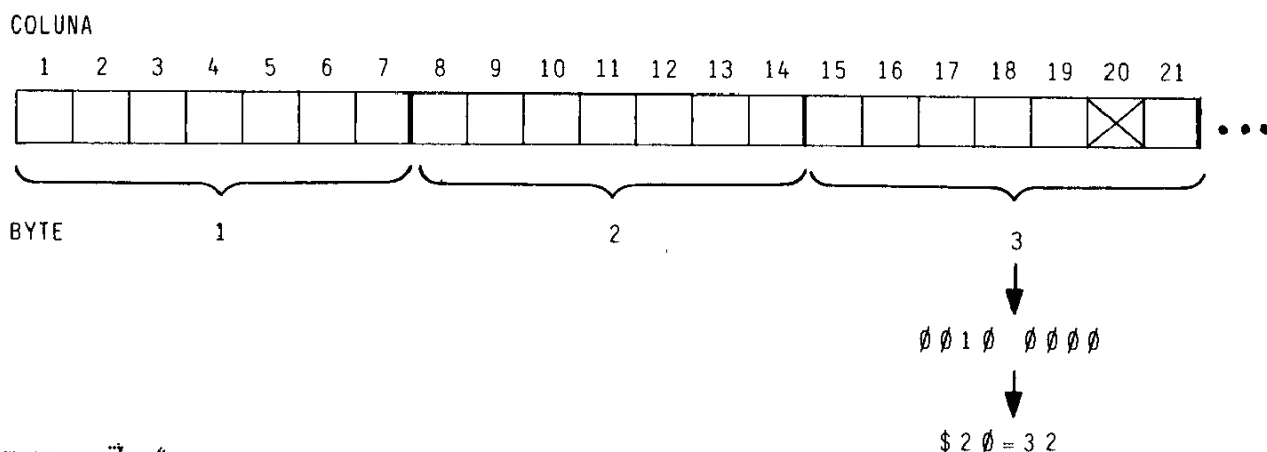


Figura 7.4

A décima linha de pontos do vídeo está na segunda linha de pontos da segunda linha de "caixas" (cada linha de caixas contém 8 linhas de pontos), e desde que cada caixa pode apresentar 7 (e não 8) pontos por linha, o vigésimo ponto está na terceira coluna, como mostra a figura 7.4. Uma vez localizado o ponto, é necessário adicionar três números para obter seu endereço: O endereço da linha, coluna, e a posição do byte no interior da caixa.

Endereço da linha:	8320	\$2080
Endereço da coluna:	2	\$ 2
Endereço da Posição	1024	\$ 400
<hr/>		
Endereço Final	9346	\$2482

Calculado o endereço, agora será necessário saber que

valor inserir nesta posição.

Observando a figura 7.4, notamos os três primeiros bytes da memória de vídeo cada um contendo 7 bits. O bit número 20 é o sexto ponto a partir da esquerda do byte. Lembrando que a formação do ponto na tela é inversa a posição do bit no byte.

Assim, a solução final para o exemplo acima será aplicar o seguinte programa:

```
>10 HGR  
>20 POKE 9346,32
```

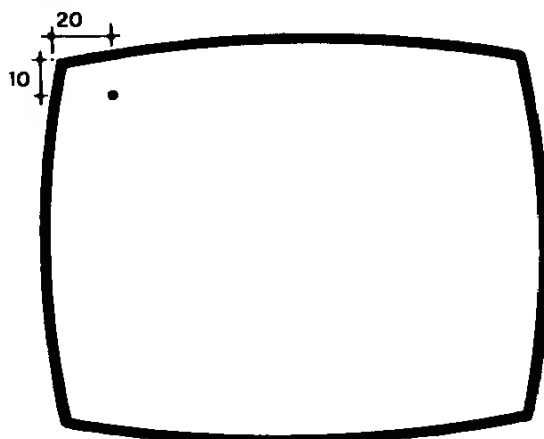


Fig. 7.5

Exemplo :

Formar a figura abaixo.



Fig.7.6 Monstrinho

A idéia inicial será sempre desenhar a figura em papel quadriculado (quando em baixa resolução) ou em papel milimetrado (quando em alta resolução). Este procedimento permite esboçar a figura facilitando o manuseio das informações. A figura apresentada acima, no caso, já veio dividida em quadrados que, na tela do computador, serão considerados como pontos.

O primeiro passo é calcular o valor de cada byte compondo o monstrinho. A tabela 7.4 apresenta o padrão dos pontos, a formação do padrão dos bits e os valores hexadecimais e decimais.

Padrão de pontos	Padrão de BIT	Valor Hexadecimal	Valor Decimal
---X---	0000 1000	\$08	08
-XXXXX-	0011 1110	\$3D	62
XXXXXXXX	0111 1111	\$7F	127
X--X--X	0100 1001	\$49	73
-XX-XX-	0011 0110	\$36	54
-X---X-	0010 0010	\$22	34
-X---X-	0010 0010	\$22	34
--X-X--	0001 0100	\$14	20

Tabela 7.4

O valor zero acrescentado a esquerda de cada byte (no padrão de bit) está para suprir o oitavo bit. O cálculo dos valores não foi difícil, agora só faltará calcular os endereços.

Note que a figura possui a largura máxima de 7 bits. Isto permitiu, como vimos, que a figura possa ser representada por apenas 7 colunas de bits. Caso a figura ocupasse mais do que 7 bits de largura total, haveria maior quantidade de bytes a serem inseridos, como veremos seguir.

Os endereços de linha e coluna dependem, neste exemplo, de qual caixa será usada para plotar o monstinho, mas o endereço para a posição na caixa está fixado pela posição de linha na figura. O endereço de posição da linha 1 é 0 (\$0), para a linha 2 é 1024 (\$400), para a linha 3 é 2048 (\$800), assim por diante como mostra o mapa de alta resolução (figura 7.2). O programa a seguir mostra as vantagens desta situação e plota um monstinho em qualquer posição da tela.

```

>10 REM INICIALIZACAO
>20 DIM RA(20)
>30 FOR I = 1 TO 20 : READ RA(I) : NEXT I
>40 DATA 8192, 8320, 8448, 8576, 8704, 8832, 8960,
9088, 8232, 8360
>50 DATA 8488, 8616, 8744, 8872, 9000, 9128, 8272,
8400, 8528, 8656
>60 REM ENTRADA DA LINHA E DA COLUNA
>70 HGR : HOME : VTAB (24)
>80 INPUT "NUMERO DE LINHA? (1-20) ";L
>90 IF L<1 OR L>20 THEN 80
>100 INPUT "NUMERO DE COLUNA? (1-40) ";C
>110 IF C<1 OR C>40 THEN 100
>120 BA = RA(L) + C - 1
>130 POKE BA , 8
>140 POKE BA + 1024, 62
>150 POKE BA + 2048, 127
>160 POKE BA + 3072, 73
>170 POKE BA + 4096, 54
>180 POKE BA + 5120, 34
>190 POKE BA + 6144, 34
>200 POKE BA + 7168, 20
>210 INPUT "VOCE QUER REPETIR ? (S OU N)";SS

```



```

>220 IF S$ <> "N" THEN 80
>230 TEXT : HOME

```

As linhas 20 a 60 dimensionam e inicializam uma área contendo os endereços das 20 linhas. A linha 70 acerta o modo gráfico de alta resolução com o cursor no fim da janela de texto. As linhas 80 até 110 aceitam os valores de entradas de linha e coluna verificando a validade dos mesmos. A linha 120 calcula os endereços de base através da soma dos endereços de linha da caixa RA(R) com o endereço de coluna (que é igual ao número de coluna menos 1). Tendo somado os endereços de linha e de coluna da caixa, falta somente somar o endereço da posição na caixa para cada byte do monstrinho. No final o endereço é calculado e o próprio valor é inserido na memória nas linhas 130 até 190. As linhas 200 e 210 dão chance de repetir o desenho e a linha 220 apaga a tela de alta resolução. A velocidade do programa poderá ser aumentada se os valores, nas linhas 130 até 190 já forem inseridos no programa diretamente através dos comandos POKE's (sem ter que perder tempo calculando qualquer coisa).

Exemplo : Através de um programa simples de BASIC, vamos montar a seguinte figura:

	A	B	C	
Linha 0		XX	XXXXXXXX	XXXX
Linha 1			X	
Linha 2	XXX		XXXXXX	X
Linha 3	X	XXXXX	XXXX	XX
Linha 4	XXX	XXX	XXXX	XX
Linha 5		XX	XXXX	XX
Linha 6		X	XXXX	XX
Linha 7			XXXXXX	X
Linha 8			X	
Linha 9			X	X
Linha 10			XXXXXXXX	XX

1

2

Fig 7.7

A primeira tarefa é de digitalizar o helicóptero. Note que há dezoito pontos de largura por onze de altura, sendo assim, o melhor será dividir o helicóptero em seis caixas para a projeção na tela. A figura acima já está dividida em seis quadrantes. As figuras abaixo mostram a digitalização de cada quadrante separadamente.

Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-----XX	0110 0000	\$60	96
-----	0000 0000	\$00	0
XXX----	0000 0111	\$07	7
X-XXXXX	0111 1101	\$7D	125
XXX-XXX	0111 0111	\$77	119
-----XX	0110 0000	\$60	96
-----X	0100 0000	\$40	64
-----	0000 0000	\$00	0

Tabela 7.5- Quadrante A1

Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
XXXXXXX	0111 1111	\$7F	127
-----X--	0001 0000	\$10	16
-XXXXXX	0111 1110	\$7E	126
XXXX----	0000 1111	\$0F	15
XXXX----	0000 1111	\$0F	15
XXXX----	0000 1111	\$0F	15
XXXX----	0000 1111	\$0F	15
-XXXXXX	0111 1110	\$7E	126

Tabela 7.6- Quadrante B1

Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
XXXX----	0000 1111	\$0F	15
-----	0000 0000	\$00	0
X-----	0000 0001	\$01	1
XX-----	0000 0011	\$03	3
-XX-----	0000 0110	\$06	6
-XX-----	0000 0110	\$06	6
XX-----	0000 0011	\$03	3
X-----	0000 0001	\$01	1

Tabela 7.7- Quadrante C1

Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0

Tabela 7.8- Quadrante A2

Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
----X--	0001 0000	\$10	16
----X--	0001 0000	\$10	16
XXXXXXX	0111 1111	\$7F	127

Tabela 7.9- Quadrante B2

Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-----	0000 0000	\$00	0
-X-----	0000 0010	\$02	2
XX-----	0000 0011	\$03	3

Tabela 7.10- Quadrante C2

A figura será projetada no canto superior esquerdo da tela.

Para facilitar o encontro dos endereços de cada linha da tela, abaixo é apresentada uma parte da tabela localizada no Apêndice C no qual cada linha já vem relacionada com sua respectiva faixa de memória:

	LINHA NÚMERO	FAIXA DE ENDEREÇOS
Primeira linha de caixas	0	8192-8231
	1	9216-9255
	2	10240-10279
	3	11264-11303
	4	12288-12327
	5	13312-13351
	6	14336-14375
	7	15360-15399
Segunda linha de caixas	8	8320-8359
	9	9344-9383
	10	10368-10407
	11	11392-11431
	12	12416-12455
	13	13440-13479
	14	14464-14503
	15	15488-15527

Tabela 7.11

A seguir, apresentamos o programa.

```

>10 REM PROGRAMA HELICOPTERO
>20 HGR
>25 REM
>30 REM QUADRANTE A1
>40 POKE 8192,96 : POKE 9216,0 : POKE 10240,7 :
      POKE 11264,125 : POKE 12288,119 : POKE
      13312,96 : POKE 14336,64 : POKE 15360,0
>45 REM

```



```

>50 REM QUADRANTE A2
>60 POKE 8320,0 : POKE 9344,0 : POKE 10368,0
>65 REM
>70 REM QUADRANTE B1
>80 POKE 8193,127 : POKE 9217,16 : POKE 10241,126
      : POKE 11265,15 : POKE 12289,15 : POKE
      13313,15 : POKE 14337,15 : POKE 15361,126
>85 REM
>90 REM QUADRANTE B2
>100 POKE 8321,16 : POKE 9345,16 : POKE 10369,127
>110 REM
>120 REM QUADRANTE C1
>130 POKE 8194,15 : POKE 9218,0 : POKE 10242,1 :
      POKE 11266,3 : POKE 12290,6 : POKE 13314,6 :
      POKE 14338,3 : POKE 15362,1
>140 REM
>150 REM QUADRANTE C2
>160 POKE 8322,0 : POKE 9346,2 : POKE 10370,3
>180 END

```

VII.3.4 Utilizando Cores em Gráficos de Alta Resolução

Boas cores são geralmente uma necessidade para a apresentação de desenhos ou gráficos em alta resolução de qualidade. As cores permitem distinguir elementos diferentes em uma mesma figura ou gráfico dando assim mais vida à tela, e tornando o resultado de saída muito mais atraente aos olhos do observador. O TK-2000 possui em seu hardware a possibilidade de produzir 6 cores na saída, incluindo o preto e o branco; sendo que, através de uma técnica chamada dithering, este número pode ser aumentado consideravelmente.

Para obter cores numa tela de alta resolução é muito simples; quase tão simples visto que sempre que um ponto é plotado, ele sempre aparece numa cor. O segredo é desenvolver a cor certa no lugar certo.

Neste ponto os conceitos de uso do mapa de memória de alta resolução já devem estar bem conhecidos. Inclusive o modo de transportar o padrão de pontos para o padrão de bits podendo ainda traduzí-los para valores hexadecimais e decimais sucessivamente. Finalmente o que resta para esta seção é relacionar cores para estes pontos.

No item anterior, foi mencionado que o bit mais significativo do byte não era apresentado na tela aludindo inclusive ao fato deste bit ter função de controle de cor. Este bit é chamado de bit de cor do byte e ele é que determina em qual grupo de cores será apresentado os 7 bits restantes. Se o bit de cor é zero, os sete bits serão apresentados em cores do grupo 1: PRETO1, AZUL, CYAN, ou BRANCO1. Se o bit de cor for um, o byte irá estar apresentado no grupo 2 de cores: PRETO2, VERMELHO, VERDE, ou BRANCO2 (Existem duas opções de preto e duas de branco, sendo que não existe diferença nenhuma entre elas). Essas cores variarão dependendo do TV ou monitor usado, mas geralmente é

possível ajustar a cor da tela de modo a projetar a cor desejada. Este assunto de bit de cor será novamente tocado adiante, entretanto digite o programa abaixo e veja o resultado:

```
>10 HGR
>20 HCOLOR=3
>30 HPLOT 1,0 TO 1,100
>40 GET R% : REM PRODUZ UMA PAUSA
>50 HCOLOR=4
>60 FOR I = 0 TO 100
>70 HPLOT 0,I
>80 NEXT I
```

De acordo com o Manual de Operação, HCOLOR=3 é BRANCO1; conseqüentemente, as linhas 10 até 30 devem plotar uma reta branca vertical a partir da coluna 1. Ao rodar o programa, uma linha vertical azul surgirá na tela. Isto ocorre porque está sendo plotado somente um ponto em cada linha, e qualquer ponto isolado sempre resulta em uma cor. A linha 40 pára o programa até que qualquer tecla seja pressionada; a linha 50 define a cor como PRETO2, pois se trata da cor preta procedente do grupo 2. As linhas 60, 70 e 80 imprimem uma linha vertical de pontos pretos na coluna 0. Visto que a coluna 0 já era ocupada por pontos pretos, seria de se esperar que nada fosse alterado, mas o que ocorre é que a linha azul acaba sendo substituída por uma linha vermelha após qualquer tecla ter sido digitada. A razão para a mudança de cor está ligado ao bit de cor. Os pontos na coluna 0 e coluna 1 são controlados pelo mesmo byte de memória da alta resolução, e plotando BRANCO1 ativou (passou para 1) o bit 1 de cada byte deixando o resto do byte nulo, incluindo o bit de cor. Plotando na coluna 0 o PRETO2 desativou os bits da coluna 0, que já estavam em 0 de qualquer maneira, mas também ativou o bit de cor de cada byte desde que PRETO2 está no grupo 2 de cores. Quando o bit de cor de cada byte passou para 1, provocou uma mudança do byte inteiro para uma cor do grupo 2 de cores; assim, o azul na coluna 1, que era uma cor do grupo 1, passou para a correspondente do grupo 2 de cores, vermelha. Se a coluna 0 tivesse sido impressa em PRETO1 ao invés de PRETO2, o bit de cor de cada byte teria permanecido em zero e nenhuma alteração de cor teria ocorrido.

Digite e rode o programa abaixo que irá ilustrar mais uma característica das cores de alta resolução.

```
>10 HGR
>20 HCOLOR = 2
>30 FOR I = 1 TO 100
>40 HPLOT I,1
>50 NEXT I
>60 REM
>70 REM APAGANDO A LINHA
>80 HCOLOR = 0
>90 FOR I = 2 TO 100 STEP 2
>100 HPLOT I,1
>110 NEXT I
```


Um exame da listagem acima irá conduzir à idéia do programa plotar 100 pontos horizontais de cor verde, e então a metade dos pontos seriam apagados. No entanto, ao rodar o programa, nota-se que a linha foi construída e depois apagada completamente. Como poderia isto ter ocorrido quando foi plotando 100 pontos e apagados somente 50? Para observar melhor este processo poderia-se incluir a linha:

```
>35 FOR J = 1 TO 500 : NEXT J : PRINT CHR$(7)
```

que indicará os pontos como se eles estivessem sendo impressos. Assim, se verificará que apenas 50 pontos foram impressos primeiro lugar. O verde está somente disponível em colunas pares (0,2,4,...), e quando o programa tenta plotar um ponto verde em uma coluna ímpar não terá efeito nenhum. O mesmo ocorre com a cor cyan. No caso das cores azul e vermelho, só serão disponíveis em colunas ímpares.

Para resumir o que foi visto no parágrafo acima, se um ponto é impresso numa coluna par, este deverá ser ou verde ou cyan, se um ponto está em uma coluna ímpar, deverá ser ou azul ou vermelho. O bit de cor seleciona qual das cores disponíveis cada ponto irá projetar. Se o bit de cor for 0 (grupo 1), nos pontos pares poderão ser projetados apenas o cyan e no caso das colunas ímpares somente aparecerá o azul; se o bit de cor for 1 (grupo 2) nas colunas pares aparecerá a cor verde, e nas colunas ímpares aparecerá a cor vermelha. Não esquecendo que sempre a primeira coluna a esquerda de cada linha é contada como 0, depois 1... até 279.

O preto é sempre plotado quando um bit passa para 0 enquanto que o branco ocorre sempre que dois bits adjacentes forem 1. Faça a seguinte experiência utilizando o comando POKE:

```
>HGR
>POKE 8192,1
>POKE 8192,2
>POKE 8192,3
```

No topo do lado esquerdo da tela aparecerá no primeiro caso, um ponto de cor cyan, no segundo, um ponto azul e finalmente um ponto branco. A tabela abaixo mostra como os valores nos comandos POKE's foram interpretados na tela:

VALOR	PADRAO DE BITS	COLUNA(S)	COR GERADA
1	0000 0001	col 0	cyan
2	0000 0010	col 1	azul
3	0000 0011	col 0 e col 1	branco

Quando os pontos ativados (passam para 1) individualmente, eles tendem a revelar suas respectivas cores. Mas quando dois pontos vizinhos estão em 1 (formando a cor branca), qualquer outro bit vizinho que passar para 1 irá projetar a cor branca.

Plotando qualquer ponto sobre ou sob outro não produzira efeito algum sobre a cor.

Finalmente, observe a resolução deste último exemplo:

[illegible]

Figura 7.8

A figura acima possuirá uma asa azul e verde e a outra vermelha e cyan sendo que o corpo e as antenas serão brancas. Note que apenas no centro do corpo e nas antenas é encontrado dois bits adjacentes iguais a um.

O primeiro passo para a construção da figura acima será de dividir adequadamente a figura num número de pontos que permita um correto posicionamento das cores. Sendo que a seguir deverá ser executado o levantamento dos bits correspondentes.

[illegible]

Setor B1 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
--XX---	1000 1100	\$8C	140
--XX---	1000 1100	\$8C	140
----XX-	1011 0000	\$B0	176
----XX-	1011 0000	\$B0	176
X-----	1000 0001	\$81	129
X-----	1000 0001	\$81	129
X-X---X	1100 0101	\$C5	197
X-X---X	1100 0101	\$C5	197

Setor C1 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
---XX---	0001 1000	\$18	24
---XX---	0001 1000	\$18	24
-XX-----	0000 0110	\$06	6
-XX-----	0000 0110	\$06	6
-----X	0100 0000	\$40	64
-----X	0100 0000	\$40	64
X---X-X	0101 0001	\$51	81
X---X-X	0101 0001	\$51	81

Setor D1 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
---X-X-	1010 1000	\$A8	168
---X-X-	1010 1000	\$A8	168
-X-X-X-	1010 1010	\$AA	170
-X-X-X-	1010 1010	\$AA	170
-X-X-X-	1010 1010	\$AA	170
-X-X-X-	1010 1010	\$AA	170
-X-X-X-	1010 1010	\$AA	170
-X-X-X-	1010 1010	\$AA	170

Setor A2 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-X-X-X-	0010 1010	\$2A	42
-X-X-X-	0010 1010	\$2A	42
---X-X-	0010 1000	\$28	40
---X-X-	0010 1000	\$28	40
-----X-	0010 0000	\$20	32
-----X-	0010 0000	\$20	32
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0

Setor B2 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213

Setor C2 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85

Setor D2 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-X-X-X-	1010 1010	\$AA	170
-X-X-X-	1010 1010	\$AA	170
-X-X----	1000 1010	\$8A	138
-X-X----	1000 1010	\$8A	138
-X-----	1000 0010	\$82	132
-X-----	1000 0010	\$82	132
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0

Setor A3 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-----X-	0010 0000	\$20	32
-----X-	0010 0000	\$20	32
----X-X-	0010 1000	\$28	40
----X-X-	0010 1000	\$28	40
----X-X-	0010 1000	\$28	40
----X-X-	0010 1000	\$28	40
----X-X-	0010 1000	\$28	40
----X-X-	0010 1000	\$28	40

Setor B3 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X-X-X	1101 0101	\$D5	213
X-X---X	1100 0101	\$C5	197
X-X---X	1100 0101	\$C5	197

Setor C3

Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X-X-X-X	0101 0101	\$55	85
X---X-X	0101 0001	\$51	81
X---X-X	0101 0001	\$51	81

Setor D3

Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-X-----	1000 0010	\$82	130
-X-----	1000 0010	\$82	130
-X-X----	1000 1010	\$8A	138
-X-X----	1000 1010	\$8A	138
-X-X----	1000 1010	\$8A	138
-X-X----	1000 1010	\$8A	138
-X-X----	1000 1010	\$8A	138
-X-X----	1000 1010	\$8A	138
-X-X----	1000 1010	\$8A	138

Setor A4

Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-----X-	0010 0000	\$20	32
-----X-	0010 0000	\$20	32
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0

Setor B4 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
X-----	1000 0001	\$81	129
X-----	1000 0001	\$81	129
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0

Setor C4 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-----X	0100 0000	\$40	130
-----X	0100 0000	\$40	130
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0

Setor D4 Padrão de pontos	Padrão de bits	Valor Hexadecimal	Valor Decimal
-X-----	1000 0010	\$82	130
-X-----	1000 0010	\$82	130
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0
-----	0000 0000	\$00	0

Abaixo é apresentado um programa BASIC que reproduzirá o desenho da borboleta:

```

>1 REM PROGRAMA BORBOLETA
>10 HGR : HOME
>20 REM
>30 REM SETOR A1
>40 DATA 10,10,42,42,42,42,42,42
>50 REM SETOR B1
>60 DATA 140,140,176,176,129,129,197,197
>70 REM SETOR C1
>80 DATA 24,24,6,6,64,64,81,81
>90 REM SETOR D1

```



```

>100 DATA 168,168,170,170,170,170,170,170
>110 REM SETOR A2
>120 DATA 42,42,40,40,32,32,0,0
>130 REM SETOR B2
>140 DATA 213,213,213,213,213,213,213,213
>150 REM SETOR C2
>160 DATA 85,85,85,85,85,85,85,85
>170 REM SETOR D2
>180 DATA 170,170,138,138,132,132,0,0
>190 REM SETOR A3
>200 DATA 32,32,40,40,40,40,40,40
>210 REM SETOR B3
>220 DATA 213,213,213,213,213,213,197,197
>230 REM SETOR C3
>240 DATA 85,85,85,85,85,85,81,81
>250 REM SETOR D3
>260 DATA 130,130,138,138,138,138,138,138
>270 REM SETOR A4
>280 DATA 32,32,0,0,0,0,0,0
>290 REM SETOR B4
>300 DATA 129,129,0,0,0,0,0,0
>310 REM SETOR C4
>320 DATA 130,130,0,0,0,0,0,0
>330 REM SETOR D4
>340 DATA 130,130,0,0,0,0,0,0
>350 FOR J = 0 TO 3
>360 FOR I = 0 TO 3
>370 FOR K = 0 TO 7
>380 T = 8488 + K * 1024 + I + J * 128
>390 READ B
>400 POKE T,B
>410 NEXT K,I,J
>420 END

```


VII.3.5 Conclusão

Esta seção terá, aparentemente, a finalidade de contrariar tudo o que já foi visto até agora. Foi mencionado (repetitivamente) que existia 280 posições na alta resolução, mas agora tentaremos explicar que, na realidade, existem apenas 40 posições, e também existem 560 posições, e algumas vezes 140. O propósito desta mudança é proporcionar outros modelos conceituais, mais claros e mais simples, para criar no modo de alta resolução. Cada modelo é baseado num diferente número de posições na tela, como será explicado a seguir.

Porque 40 colunas ?

A explicação do modelo de 40 posições é o mais simples desde que já se tenha trabalhado com ele, talvez inconscientemente. Para processar qualquer ponto, é necessário inevitavelmente tratar com o byte que contém cada ponto, e a largura da tela é constituída por apenas 40 bytes.

No item anterior, quando foi desenvolvida e impressa uma figura em alta resolução, todo o trabalho foi realizado em incrementos de um byte e impresso com a consideração da existência de apenas 40 "caixas" na largura da tela onde foi desenhada a figura.

Porque 280 colunas ?

O modelo de 280 posições ainda é válido. Cada um dos 40 bytes projetam 7 pontos, produzindo 280 pontos na tela. O conceito de 280 pontos é útil quando se está desenhando figuras, mas quando se está iniciando a digitalizar os dados, naturalmente se adotará o modelo de 40 colunas. Vide os exemplos anteriores.

Porque 140 colunas ?

Quando se trabalha com cores em alta resolução, a maioria dos programadores considera a largura da tela como sendo de 140 colunas. Como foi visto, para se plotar uma reta verde, por exemplo, era necessário que deixássemos uma coluna vazia entre dois pontos, o mesmo acontecia caso fosse necessário imprimir um ponto branco, era necessário que dois pontos adjacentes estivessem em 1. Assim, desde que a unidade de cada cor é de dois pontos, a tela inteira poderá ser considerada como que com 140 colunas.

Além disso, quando uma determinada figura é plotada em diversas posições, as cores mudarão dependendo de onde elas foram impressas. Para que se possa manter as cores originais, qualquer ponto que estava originalmente numa coluna par deverá ser plotada numa coluna par, a mesma idéia para as cores ímpares. Para efetuar então qualquer movimento na figura, ela deverá se mover sempre em múltiplos de dois pontos por vez, por esta razão, há apenas 140 posições possíveis através da tela.

Porque 560 colunas ?

Se bem que até agora só se falou numa resolução máxima de 280 pontos, existem normalmente 560 pontos disponíveis na largura da tela de alta resolução. Para provar a afirmação acima, digite as seguintes instruções.

>HGR

A Tabela 7.11 mostra os endereços hexadecimais do primeiro byte de cada uma das 14 linhas da tela, o padrão de pontos e de bits de 14 valores diferentes que serão colocados nestas posições.

Endereco Hexadecimais	Valor Hexadecimal	Padrao de bits	Padrao de pontos
2000	01	0000 0001	X-----
2400	81	1000 0001	X-----
2800	02	0000 0010	--X-----
2C00	82	1000 0010	--X-----
3000	04	0000 0100	---X-----
3400	84	1000 0100	---X-----
3800	08	0000 1000	----X-----
3C00	88	1000 1000	----X-----
2080	10	0001 0000	-----X---
2480	90	1001 0000	-----X---
2880	20	0010 0000	-----X--
2C80	A0	1010 0000	-----X--
3080	40	0100 0000	-----X
3480	C0	1100 0000	-----X

Tabela 7.11

Insira agora os valores na memória digitando:

```
@2000:01
@2400:81
@2800:02
.
.
.
@3480:C0
```

Observando o padrão de pontos, era de se esperar um resultado de sete pares de pontos, com um ponto de cada par exatamente acima do outro.

No entanto, quando os valores foram entrando na memória, o que se formou foi uma linha diagonal; os pares de pontos não estavam empilhados como esperado, ainda que o mesmo ponto tenha ficado igual a 1 em cada byte do par. Este fenômeno baseia-se no fato que o primeiro byte de cada par estava numa cor do grupo 1, e o segundo byte estava numa cor do grupo 2, determinado pelo bit de cor no lado esquerda do byte.

resolução são capazes de produzir somente preto, branco, azul, verde, vermelho e cyan; mas usando imaginação e criatividade, será possível produzir "artificialmente" colorações extras através de um processo chamado "dithering" (excitação).

Rode o programa abaixo:

```
>10 TE = 200 : REM ATRASO
>20 TA = 50 : REM TAMANHO
>30 HGR
>40 FOR C1 = 1 TO 7
>50 FOR C2 = 1 TO 7
>60 FOR Y = 0 TO TA STEP 2
>70 HCOLOR C1
>80 HPLLOT 0,Y TO TA,Y
>90 HCOLOR C2
>100 HPLLOT 0,Y+1 TO TA,Y+1
>110 NEXT Y
>120 FOR I = 1 TO TE : NEXT I
>130 NEXT C2
>140 NEXT C1
>150 END
```

O resultado será um retângulo no canto superior esquerdo da tela que continuamente muda de cor, às vezes apresenta apenas sombras, às vezes apresenta uma das 6 cores básicas. É possível alterar a velocidade das mudanças variando a TE na linha 30 e alterar o tamanho do retângulo variando TA na linha 40.

A idéia principal do programa é plotar pares de linhas horizontais em duas cores alternadas (C1 e C2), então muda-se uma ou as duas cores e inicia-se novamente. Durante os vários ciclos do programa acima, é possível ver as cores básicas (azul, verde, vermelho, cyan, branco e preto - mas também algumas outras cores. As cores extras foram, então, geradas pela mistura de duas cores distintas gerando assim uma terceira.

Existe uma enorme quantidade de variações que permitem produzir uma rica gama de cores.

Contudo, deve ser observado que, pode inadvertidamente ser mudado um ponto de um grupo de cores para outro ao se tentar imprimir cores de grupos de cores diferentes dentro de um mesmo byte. Portanto, há certas combinações que não deverão produzir os efeitos esperados. Assim mesmo, não se deve esquecer que, se dois pontos ficarem adjacentes, o resultado será branco.

Um cuidadoso planejamento permitirá obter resultados artisticamente perfeitos. Experimente.

Este capítulo apresenta os principais dados externos e internos que devem ser registrados para o uso normal de seu TK-2000.

VIII.1 Diagrama de blocos

A CPU do seu TK-2000 COLOR COMPUTER é o 6502 que possui 16 linhas de endereços e 8 linhas de dados acessíveis no conector de expansão.

A CPU opera numa frequência de 1,021 MHz, obtida de um circuito de temporização a partir de 14,3 MHz. Este circuito também é responsável pela geração de sinais de sincronismo para o decodificador de endereços, para as RAM's dinâmicas e para o controlador de vídeo.

O software residente do seu TK-2000 está contido em 16 Kbytes de ROM, permitindo o controle de operações dos programas em BASIC e em linguagem de máquina.

O circuito decodificador de endereço é o responsável pela seleção do(s) bloco(s) que se comunicam com a CPU. Nele também existe um multiplexador que gera o endereçamento das RAM's dinâmicas; este endereço é selecionado entre o barramento de endereços da CPU e o endereço gerado pelo controlador de vídeo.

O teclado é composto por uma matriz interligada ao barramento de dados por dois buffer/interfaces.

O primeiro buffer/interface armazena uma informação gerada pela CPU, que provoca uma resposta pelo teclado lida pela CPU através do segundo buffer/interface.

A interface de impressora está ligada aos mesmos buffer/interfaces do teclado e o joystick está em paralelo com as teclas de cursor, FIRE e letra L.

A interface de cassette está associada a uma linha de dados, permitindo assim o reconhecimento pela CPU da informação lida.

Abaixo é apresentado o diagrama de blocos do TK-2000 cuja finalidade é permitir que se obtenha uma clara visão do funcionamento de seu sistema.

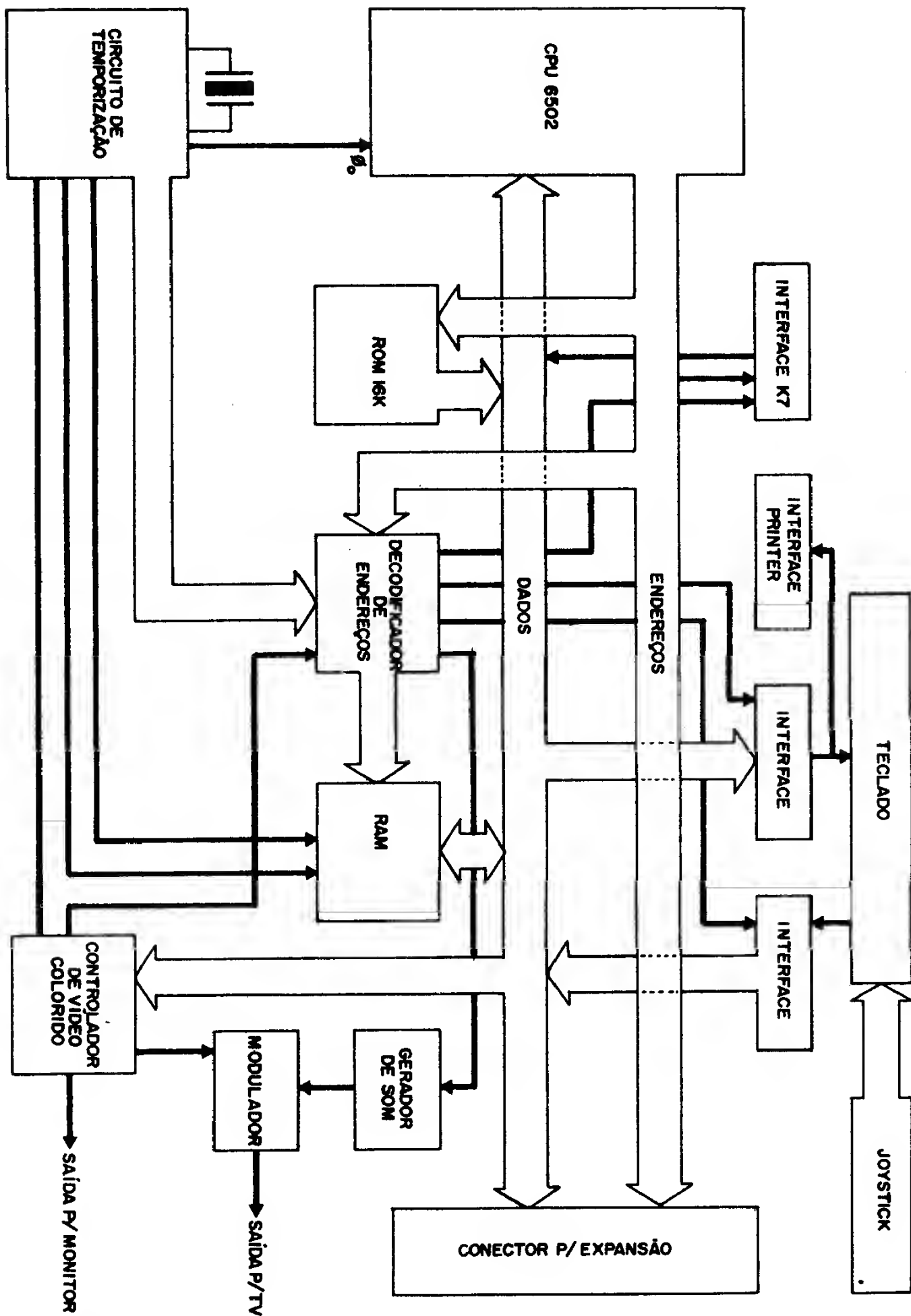


Fig. VIII.1

VIII.2 0 Microprocessador

Modelo:	6502
Número de instruções:	56
Modos de endereçamento:	13
Acumuladores:	1 (A)
Registros de Index:	2 (X,Y)
Outros registros:	Stack Pointer (S) Status Register (P) Program Counter (PC)
Stack:	256 bytes, fixado na página \$01
Bandeiras do Status:	S (signal) C (carry) V (overflow) Z (zero)
Outras bandeiras:	I (Interrupt disable) D (Decimal mode) B (Break)
Interrupções:	2 (IRQ, NMI)
Reset:	1 (RES)
Faixa de endereçamento:	2^{16} (64 K) posições
Linha de endereços:	16 bits, paralelos
Linha de dados:	8 bits, paralelos e bidirecionais
Tensão de trabalho:	+ 5 Volts
Potência dissipada:	.25 watt
Frequência de clock: no TK-2000	1.021 MHz

VIII.3. Conector de Saída - Expansão

A seguir é apresentada a pinagem no conector de saída cuja finalidade será explicada adiante.

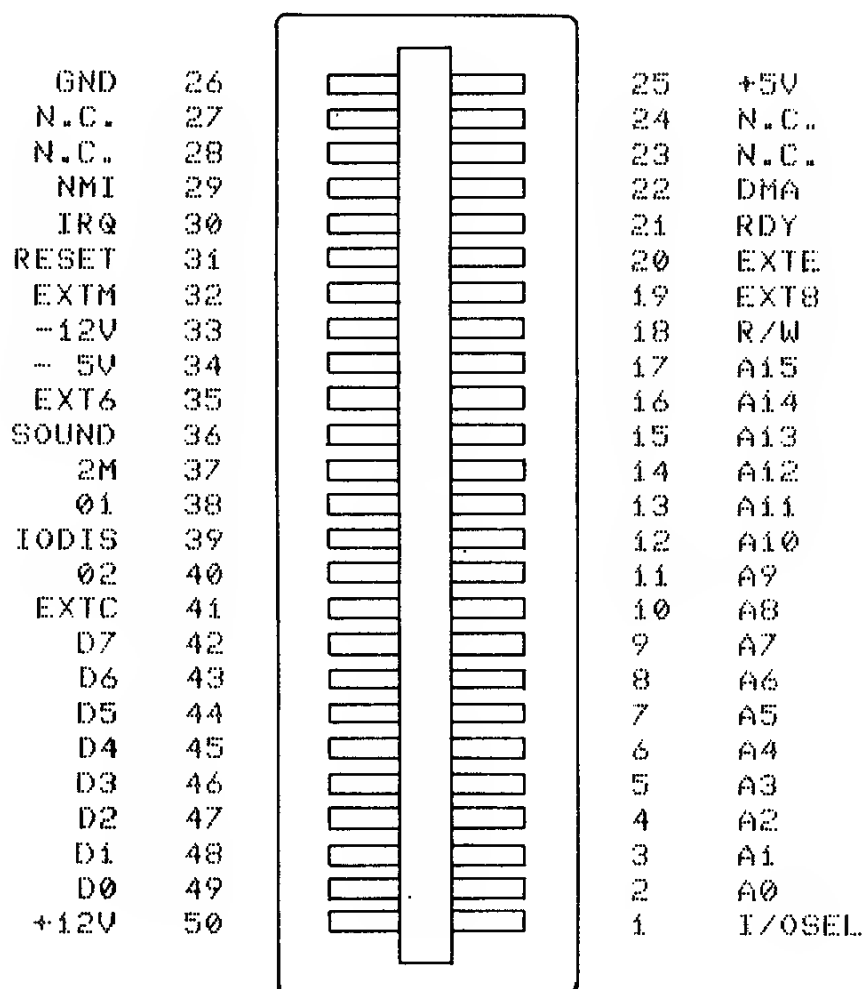


Fig. VIII.2

Descrição dos Sinais do Conector de Expansão

Pino:	Nome:	Função:
1	I/OSEL (OUTPUT)	Esta linha normalmente está em nível 1, somente passará para 0 quando o microprocessador estiver acessando qualquer endereço entre \$C000 e \$C0FF. Este sinal torna-se ativo durante o período 02.
2-17	A0-A15 (OUTPUT)	Linha bufferizada de endereços. Os endereços contidos nestas linhas tornam-se válidos durante o período 01 e mantêm-se válidos durante 02.

18	R/W (OUTPUT)	Sinal READ/WRITE. Torna-se válido ao mesmo tempo que o barramento de endereços. Tem o valor 0 num ciclo de escrita e 1 num ciclo de leitura.
19	EXTE B (OUTPUT)	Este sinal está disponível para selecionar qualquer dispositivo externo e é ativo (em zero) na área de memória equivalente à faixa de endereços de \$8000 até \$9FFF.
20	EXTE (OUTPUT)	Este sinal está disponível para selecionar qualquer dispositivo externo e é ativo (em zero) na área de memória equivalente à faixa de endereços de \$E000 até \$FFFF.
21	RDY (INPUT)	Esta é a entrada do 6502. Levando esta linha para o nível 0, o microprocessador deterá sua execução, e o barramento de dados apresentará o conteúdo do Program Counter.
22	DMA (INPUT)	Mantendo esta linha em nível baixo desabilita-se o barramento de endereços do 6502 e paraliza-se o processador. Esta linha é normalmente mantida em nível 1 através de um resistor de 1 K ligado a +5v.
23	N.C.	Não conectado
24	N.C.	Não conectado
25	+5 V	Tensão de alimentação.
26	GND	Terra do sistema
27	N.C.	Não conectado
28	N.C.	Não conectado
29	NMI (INPUT)	Interrupção não mascarada. Quando este pino é colocado em nível 0 o TK-2000 inicia um ciclo de interrupção e salta para uma rotina localizada no endereço \$3FB.

30	IRQ (INPUT)	Requisição de interrupção mascarada. Quando esta linha é levada a nível 0 o TK-2000 inicia o ciclo de interrupção somente se a bandeira I estiver desativada. Caso positivo, o processador irá pular para uma rotina cujo endereço está guardado nas posições \$3FE e \$3FF.
31	RESET (INPUT)	Quando esta linha é posta em nível 0 o processador inicia um ciclo de RESET.
32	EXTM (INPUT)	Este sinal desabilita a ROM e a RAM interna quando for aplicado um nível lógico 0 (zero).
33	-12 v	Saída de -12 V para alimentação do periférico.
34	- 5 v	Saída de - 5 v para alimentação do periférico.
35	EXT6 (OUTPUT)	Este sinal está disponível para selecionar dispositivos externos e está ativo (em zero) na área de memória equivalente à faixa de endereços de \$6000 até \$7FFF.
36	SOUND	Neste pino pode-se somar ao gerador de som do TK-2000 um sinal de som que irá sair pelo alto falante da TV, ou retirar o sinal sonoro do seu TK-2000 para ser colocado em um amplificador ou equivalente. A impedância de saída é de 5K6 e no caso de se entrar com sinal de som, deve-se colocar um resistor de 5K6 em série para uma excursão máxima de + 5V em relação ao terra (pino 26).
37	2 MHz (OUTPUT)	Saída de clock de 2.042 MHz.
38	01 OUTPUT	Fase 1 da saída de clock do microprocessador.
39	IODIS (INPUT)	Este sinal desabilita o funcionamento dos I/O, isto é, a área de memória \$C000 até \$C0FF fica desabilitada.

40	02 (OUTPUT)	Fase 2 da saída de clock do microprocessador.
41	EXTC (OUTPUT)	Este sinal está disponível para selecionar dispositivos externo e está ativo (em zero) na área de memória equivalente à faixa de endereços de C000 até \$DFFF. Lembre-se que a área \$C000 até \$C0FF é usada para I/O.
42-49	D0-D7 INPUT/OUTPUT	Barramento de dados.
50	+12 V	Saída de +12 v..

Este conector permite a interligação de variados interfaces de expansão no TK-2000, como por exemplo: interface de disco, interface serial RS-232, etc.

Esta porta de expansão garante, que não exista obsolescência e que novidades possam ser incorporadas ao TK-2000 COLOR.

VIII.4 Conectores de interface de cassette

Os quatro conectores localizados na traseira de seu TK-2000 foram projetados para trabalharem com todo tipo de gravador cassette existente em nosso mercado.

As duas saídas de controle de gravador são controladas através do comando MOTOR e comportam-se como uma chave programada. Através de um comando de software a saída (A ou B) é considerada como um curto (ligando o(s) gravador(es)) ou senão em aberto (desligando o(s) gravador(es)).

O pino de saída de dados deve, como consta no Manual de Operação, estar ligado à saída EAR de seu gravador cassette. A tensão de entrada deverá ser 1 Volt pico-a-pico (nominal). A impedância de entrada é de 12 K Ohm.

O pino de saída de dados (MIC) deve ser ligado à entrada de microfone do gravador. A tensão de saída é de 25 mv e a impedância de saída é de 100 Ohm.

As informações armazenadas ou lidas do gravador, são transmitidas em forma serial, ou seja, bit a bit. Ao transmitir, o computador decompõe os bytes em bits, e ao efetuar a leitura, compõe os bits lidos em bytes.

O canal de saída de informações para o gravador é acessado pelo registro CASOUT no endereço \$C020. O modo de operação é similar ao do canal de som (ver Cap. VI).

O canal de leitura de informações de gravador é efetuado através do bit D7 do registro KBIN (endereço \$C010).

VIII.5 Conector de Joystick

A saída de joystick do TK-2000 está ligada em paralelo a algumas linhas do teclado representando as teclas L, FIRE, ↑, ↓, → e ←. O conector de joystick apresenta a seguinte pinagem:

Conector Macho do TK-2000 para Joystick

Pino	Sinal	Comentário
1	D7	KBOUT
2	D4	KBIN
3	N.C.	
4	N.C.	
5	D6	KBOUT
6	D0	KBIN
7	D3	KBOUT
8	D5	KBOUT
9	D4	KBOUT

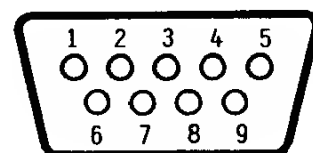


Fig. VIII.3

VIII.6 Interface de Impressora

A saída para impressora se compõe basicamente do seguinte grupo de sinais: linhas de saída de dados acionadas diretamente pelo registro KBOUT (end. \$C000), linha de saída STROBE e linha de entrada BUSY. Estes sinais aparecem na figura VIII.4

Pela linha BUSY, a impressora informa ao computador se está em condições de receber dados; entretanto pelo sinal de STROBE o computador informara à impressora que os dados que está lendo são confiáveis.

O processo de comunicação entre o computador e a impressora é como segue: o computador verifica a linha BUSY e aguarda a habilitação para o envio dos dados. Neste caso os dados devem ser colocados no registro KBOUT e posteriormente o sinal de STROBE é ativado durante alguns microsegundos, tempo requerido pela impressora.

A leitura da linha BUSY é feita lendo o bit D6 do registro KBIN. Lembre que os bits de D0 até D5 são usados para a leitura do teclado. Um trecho de programa como:

```
        SNDCHR      BIT    $C010
                   BUS    SNDCHR
```

fará que o 6502 aguarde a habilitação da linha BUSY.

Se os dados já se encontram no acumulador para enviá-los, precisamos simplesmente de um:

```
        STA    $C000
        NOP
```

O NOP atrasa a execução do próximo comando que irá colocar o sinal de STROBE, previamente em 1, em zero. O acesso ao registro STR0 (endereço \$C058) efetuará esta função; posteriormente um acesso ao registro STR1 (endereço \$C059) voltará o nível de saída em 1 lógico.

O seguinte trecho de programa irá gerar o pulso de STROBE.

```
        LDA    $C058
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        LDA    $C059
        RTS
```

Observe que foi gerado um atraso de 20 períodos de

clock, aproximadamente 20 microsegundos.

A sequência de dados depende do tipo de impressora e alguns sinais podem ter características um pouco diferentes, mas a grande facilidade na manipulação dos sinais permite através de simples programação mudar o comportamento da saída de impressora.

MARCA PRETA

D7	D6	D5	D4	D3	D2	D1	D0
15	13	11	9	7	5	3	1
16	14	12	10	8	6	4	2
N	B	G	G	G	G	G	S
C	U	N	N	N	N	N	T
	S	D	D	D	D	D	R
	Y						B

Fig. VIII.4

VIII.7 - Fonte de alimentação

O conector da fonte de alimentação do TK-2000 está provido de 4 tensões diferentes: +5 , +12 , -12 e 0 Volts estão distribuídas no conector da seguinte forma:

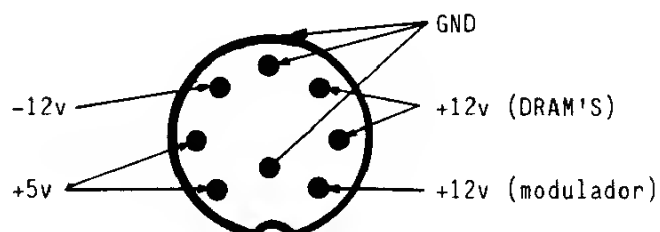


Fig. VIII.5

APÊNDICE A

INSTRUÇÕES DO MICROPROCESSADOR 6502

ADC	Adiciona com Carry memória ao acumulador
AND	Realiza operação "AND" de memória com o acumulador
ASL	Desloca para a esquerda em um bit (memória ou acumulador)
BCC	Salta quando o Carry está em zero
BCS	Salta quando o Carry está em um
BEQ	Salta quando o resultado é zero
BIT	Testa os bits da memória com o acumulador, posicionando os bits de status
BMI	Salta quando o resultado é menor que zero
BNE	Salta quando o resultado não é zero
BPL	Salta quando o resultado não é negativo
BRK	Força um BREAK
BVC	Salta quando Overflow está em zero
BVS	Salta quando Overflow está em um
CLC	Zera o flag de Carry
CLD	Zera o modo decimal
CLI	Zera o bit da máscara de interrupção (habilita interrupção)
CLV	Zera o flag de overflow
CMP	Compara a memória com o acumulador
CPX	Compara a memória com o Index X
CPY	Compara a memória com o Index Y
DEC	Decrementa a memória em uma unidade
DEX	Decrementa o Index X em uma unidade
DEY	Decrementa o Index Y em uma unidade
EOR	"EXCLUSIVE-OR" da memória com o acumulador
INC	Incrementa a memória em uma unidade
INX	Incrementa o Index X em uma unidade
INY	Incrementa o Index Y em uma unidade
JMP	Pula para novo endereço
JSR	Pula para nova subrotina, salvando o endereço de retorno
LDA	Carrega o acumulador com a memória
LDX	Carrega o Index X com a memória
LDY	Carrega o Index Y com a memória
LSR	Desloca para direita em um bit (memória ou acumulador)
NOP	Não opera
ORA	Realiza operação OR da memória com o acumulador
PHA	Empurra o valor do acumulador para o Stack
PHP	Empurra o status do processador para o Stack
PLA	Retorna o valor do acumulador do Stack
PLP	Retorna o valor do status do processador do Stack
ROL	Rotação de um bit para a esquerda (memória ou acumulador)
ROR	Rotação de um bit para a direita (memória ou acumulador)
RTI	Retorna de uma interrupção
RTS	Retorna de uma subrotina
SBC	Subtrai com Carry a memória do acumulador
SEC	Faz o Carry igual a um (1)

SED	Habilita o modo decimal
SEI	Faz o bit da máscara de interrupção igual a um (desabilita interrupção)
STA	Armazena o valor do acumulador na memória
STX	Armazena o valor do Index X na memória
STY	Armazena o valor do Index Y na memória
TAX	Transfere o acumulador para o Index X
TAY	Transfere o acumulador para o Index Y
TSX	Transfere o Stack Pointer para o Index X
TXA	Transfere o Index X para o acumulador
TXS	Transfere o Index X para o Stack Pointer
TYA	Transfere o Index Y para o acumulador

MODELO DE PROGRAMAÇÃO DO 6502

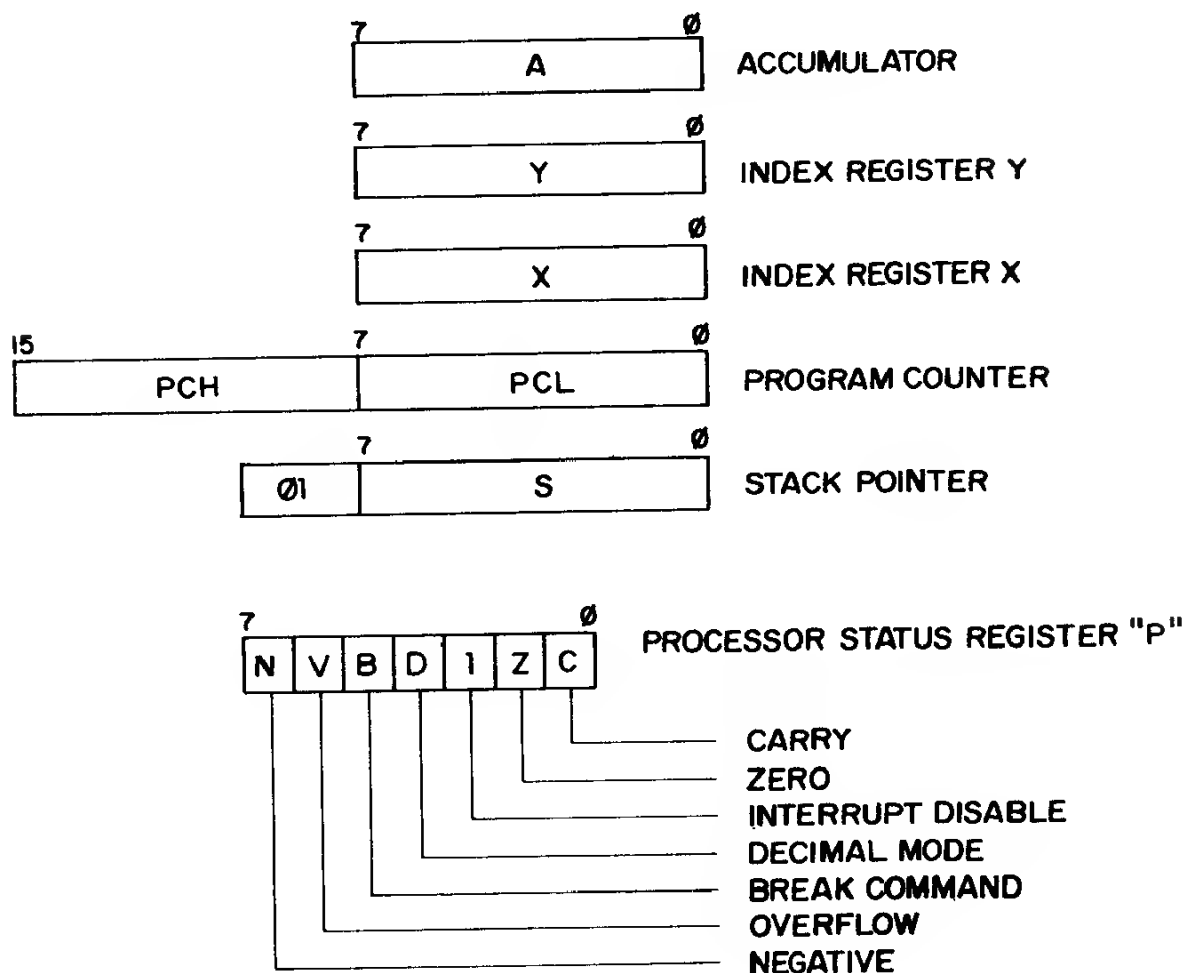


Fig. A.0

NOTAÇÃO UTILIZADA NESTE SUMÁRIO

A	Acumulador
X,Y	Registros Index
M	Memória
C	Carry
P	Registrador do status do processador (Status register)
S	Stack Pointer
+	Adição
^	AND lógico
-	Subtração
⊕	OR EXCLUSIVE lógico
↑	Transfere do Stack
↓	Transfere para o Stack
->	Transfere para
<-	Transfere para
∨	OR lógico
PC	Contador do programa (Program Counter)
PCH	Bit mais significativo do contador do programa
PCL	Bit menos significativo do contador do programa
OPER	Operando
#	Modo de endereçamento imediato

Para a tabela das páginas 87 e 88:

/	Bit afetado de acordo com a operação
-	Permanece inalterado

Códigos das instruções

Nome	Operação	Modo de Endereçamento	Formato Assembler	Cod. Hexa	Num bytes
ADC	A+M+C -->A,C	Imediato	ADC #Oper	69	2
		Pág. Zero	ADC Oper	65	2
		Pág. Zero,X	ADC Oper,X	75	2
		Absoluto	ADC Oper	6D	3
		Absoluto,X	ADC Oper,X	7D	3
		Absoluto,Y	ADC Oper,Y	79	3
		(Indireto,X)	ADC (Oper,X)	61	2
		(Indireto),Y	ADC (Oper),Y	71	2
AND	A ^ M --> A	Imediato	AND #Oper	29	2
		Pág. Zero	AND Oper	25	2
		Pág. Zero,X	AND Oper,X	35	2
		Absoluto	AND Oper	2D	3
		Absoluto,X	AND Oper,X	3D	3
		Absoluto,Y	AND Oper,Y	39	3
		(Indireto,X)	AND (Oper,X)	21	2
		(Indireto),Y	AND (Oper),Y	31	2
ASL	(Veja fig. A-1)	Acumulador	ASL A	0A	1
		Pág. Zero	ASL Oper	06	2
		Pág. Zero,X	ASL Oper,X	16	2
		Absoluto	ASL Oper	0E	3
		Absoluto,X	ASL Oper,X	1E	3
BCC	Salta qdo C = 0	Relativo	BCC Oper	90	2
BCS	Salta qdo C = 1	Relativo	BCS Oper	B0	2
BEQ	Salta qdo Z = 1	Relativo	BEQ Oper	F0	2
BIT	A^M,M7-->N M6 -->V	Pág. Zero	BIT* Oper	24	2
		Absoluto	BIT* Oper	2C	3
BMI	Salta qdo N = 1	Relativo	BMI Oper	30	2
BNE	Salta qdo Z = 0	Relativo	BNE Oper	D0	2

Nome	Operação	Modo de Endereçamento	Formato Assembler	Cod. Hexa	Num bytes
BPL	Salta qdo N = 0	Relativo	BPL Oper	10	2
BRK	PC+2 ↓ P ↓	Implícito	BRK	00	1
BVC	Salta qdo V = 0	Relativo	BVC Oper	50	2
BVS	Salta qdo V = 1	Relativo	BVS Oper	70	2
CLC	0 → C	Implícito	CLC	18	1
CLD	0 → D	Implícito	CLD	D8	1
CLI	0 → I	Implícito	CLI	58	1
CLV	0 → V	Implícito	CLV	B8	1
CMP	A - M	Imediato	CMP #Oper	C9	2
		Pág. Zero	CMP Oper	C5	2
		Pág. Zero, X	CMP Oper, X	D5	2
		Absoluto	CMP Oper	CD	3
		Absoluto, X	CMP Oper, X	DD	3
		Absoluto, Y	CMP Oper, Y	D9	3
		(Indireto, X)	CMP (Oper, X)	C1	2
		(Indireto), Y	CMP (Oper), Y	D1	2
CPX	X - M	Imediato	CPX #Oper	E0	2
		Pág. Zero	CPX Oper	E4	2
		Absoluto	CPX Oper	EC	3
CPY	Y - M	Imediato	CPY #Oper	C0	2
		Pág. Zero	CPY Oper	C4	2
		Absoluto	CPY Oper	CC	3
DEC	M - 1 → M	Pág. Zero	DEC Oper	C6	2
		Pág. Zero, X	DEC Oper, X	D6	2
		Absoluto	DEC Oper	CE	3
		Absoluto, X	DEC Oper, X	DE	3
DEX	X - 1 → X	Implícito	DEX	CA	1

Nome	Operação	Modo de Endereçamento	Formato Assembler	Cod. Hexa	Num bytes
DEY	Y - 1 -->Y	Implícito	DEY	88	1
EOR	A V M -->A	Imediato	EOR #Oper	49	2
		Pág. Zero	EOR Oper	45	2
		Pág. Zero,X	EOR Oper,X	55	2
		Absoluto	EOR Oper	4D	3
		Absoluto,X	EOR Oper,X	5D	3
		Absoluto,Y	EOR Oper,Y	59	2
		(Indireto,X)	EOR (Oper,X)	41	2
		(Indireto),Y	EOR (Oper),Y	51	2
INC	M + 1 -->M	Pág. Zero	INC Oper	E6	2
		Pág. Zero,X	INC Oper,X	F6	2
		Absoluto	INC Oper	EE	3
		Absoluto,X	INC Oper,X	FE	3
INX	X + 1 -->X	Implícito	INX	E8	1
INY	Y + 1 -->Y	Implícito	INY	C8	1
JMP	(PC+1) -->PCL	Absoluto	JMP Oper	4C	3
	(PC+2) -->PCH	Indireto	JMP (Oper)	6C	3
JSR	PC+2 ↓ (PC+1) -->PCL (PC+2) -->PCH	Absoluto	JSR Oper	20	3
LDA	M -->A	Imediato	LDA #Oper	A9	2
		Pág. Zero	LDA Oper	A5	2
		Pág. Zero,X	LDA Oper,X	B5	2
		Absoluto	LDA Oper	AD	3
		Absoluto,X	LDA Oper,X	BD	3
		Absoluto,Y	LDA Oper,Y	B9	3
		(Indireto,X)	LDA (Oper,X)	A1	2
		(Indireto),Y	LDA (Oper),Y	B1	2
LDX	M -->X	Imediato	LDX #Oper	A2	2
		Pág. Zero	LDX Oper	A6	2
		Pág. Zero,Y	LDX Oper,Y	B6	2
		Absoluto	LDX Oper	AE	3
		Absoluto,Y	LDX Oper,Y	BE	3
LDY	M -->Y	Imediato	LDY #Oper	A0	2
		Pág. Zero	LDY Oper	A4	2
		Pág. Zero,X	LDY Oper,X	B4	2
		Absoluto	LDY Oper	AC	3
		Absoluto,X	LDY Oper,X	BC	3

Nome	Operação	Modo de Endereçamento	Formato Assembler	Cod. Hexa	Num bytes
LSR	(Veja fig. A-2)	Acumulador	LSR A	4A	1
		Pág. Zero	LSR Oper	46	2
		Pág. Zero,X	LSR Oper,X	56	2
		Absoluto	LSR Oper	4E	3
		Absoluto,X	LSR Oper,X	5E	3
NOP	Não opera	Implícito	NOP	EA	1
ORA	A V M -->A	Imediato	ORA #Oper	09	2
		Pág. Zero	ORA Oper	05	2
		Pág. Zero,X	ORA Oper,X	15	2
		Absoluto	ORA Oper	0D	3
		Absoluto,X	ORA Oper,X	1D	3
		Absoluto,Y	ORA Oper,Y	19	3
		(Indireto,X)	ORA (Oper,X)	01	2
		(Indireto),Y	ORA (Oper),Y	11	2
PHA	A ↓	Implícito	PHA	48	1
PHP	P ↓	Implícito	PHP	08	1
PLA	A ↑	Implícito	PLA	68	1
PLP	P ↑	Implícito	PLP	28	1
ROL	(Veja fig. A-2)	Acumulador	ROL A	2A	1
		Pág. Zero	ROL Oper	26	2
		Pág. Zero,X	ROL Oper,X	36	2
		Absoluto	ROL Oper	2E	3
		Absoluto,X	ROL Oper,X	3E	3
ROR	(Veja fig. A-3)	Acumulador	ROR A	6A	1
		Pág. Zero	ROR Oper	66	2
		Pág. Zero,X	ROR Oper,X	76	2
		Absoluto	ROR Oper	6E	3
		Absoluto,X	ROR Oper,X	7E	3
RTI	P ↑ C ↑	Implícito	RTI	40	1
RTS	PC ↑ PC+1 -->PC	Implícito	RTS	60	1

Nome	Operação	Modo de Endereçamento	Formato Assembler	Cod. Hexa	Num bytes
SBC	A-M- \overline{C} --->A	Imediato	SBC #Oper	E9	2
		Pág. Zero	SBC Oper	E5	2
		Pág. Zero,X	SBC Oper,X	F5	2
		Absoluto	SBC Oper	ED	3
		Absoluto,X	SBC Oper,X	FD	3
		Absoluto,Y	SBC Oper,Y	F9	3
		(Indireto,X)	SBC (Oper,X)	E1	2
		(Indireto),Y	SBC (Oper),Y	F1	2
SEC	1 --->C	Implícito	SEC	38	1
SED	1 --->D	Implícito	SED	F8	1
SEI	1 --->I	Implícito	SEI	78	1
STA	A --->M	Pág. Zero	STA Oper	85	2
		Pág. Zero,X	STA Oper,X	95	2
		Absoluto	STA Oper	8D	3
		Absoluto,X	STA Oper,X	9D	3
		Absoluto,Y	STA Oper,Y	99	3
		(Indireto,X)	STA (Oper,X)	81	2
		(Indireto),Y	STA (Oper),Y	91	2
STX	X --->M	Pág. Zero	STX Oper	86	2
		Pág. Zero,Y	STX Oper,Y	96	2
		Absoluto	STX Oper	8E	3
STY	Y --->M	Pág. Zero	STY Oper	84	2
		Pág. Zero,X	STY Oper,X	94	2
		Absoluto	STY Oper	8C	3
TAX	A --->X	Implícito	TAX	AA	1
TAY	A --->Y	Implícito	TAY	AB	1
TSX	S --->X	Implícito	TSX	BA	1
TXA	X --->A	Implícito	TXA	8A	1
TXS	X --->S	Implícito	TXS	9A	1
TYA	Y --->A	Implícito	TYA	98	1

Função	Bandeiras do Status Register						
	N	Z	C	I	D	V	
ADC	/	/	/	-	-	/	
AND	/	/	-	-	-	-	
ASL	/	/	/	-	-	-	
BCC	-	-	-	-	-	-	
BCS	-	-	-	-	-	-	
BEQ	-	-	-	-	-	-	
BIT	M7	/	-	-	-	M6	
BMI	-	-	-	-	-	-	
BNE	-	-	-	-	-	-	
BPL	-	-	-	-	-	-	
BRK	-	-	-	1	-	-	
BVC	-	-	-	-	-	-	
BVS	-	-	-	-	-	-	
CLC	-	-	-	0	-	-	
CLD	-	0	-	-	-	-	
CLI	-	-	-	0	-	-	
CLV	0	-	-	-	-	-	
CMP	/	/	/	-	-	-	
CPX	/	/	/	-	-	-	
CPY	/	/	/	-	-	-	
DEC	/	/	-	-	-	-	
DEX	/	/	-	-	-	-	
DEY	/	/	-	-	-	-	
EOR	/	/	-	-	-	-	
INC	/	/	-	-	-	-	
INX	/	/	-	-	-	-	
INY	/	/	-	-	-	-	
JMP	-	-	-	-	-	-	
JSR	-	-	-	-	-	-	
LDA	/	/	-	-	-	-	
LDX	/	/	-	-	-	-	
LDY	/	/	-	-	-	-	
LSR	0	/	/	-	-	-	
NOP	-	-	-	-	-	-	
ORA	/	/	-	-	-	-	
PHA	-	-	-	-	-	-	
PHP	-	-	-	-	-	-	
PLA	/	/	-	-	-	-	
PLP	do Stack						
ROL	/	/	/	-	-	-	
ROR	/	/	/	-	-	-	
RTI	do Stack						
RTS	-	-	-	-	-	-	
SBC	/	/	/	-	-	/	
SEC	-	-	1	-	-	-	
SED	-	-	-	-	1	-	
SEI	-	-	-	1	-	-	
STA	-	-	-	-	-	-	
STX	-	-	-	-	-	-	
STY	-	-	-	-	-	-	
TAX	/	/	-	-	-	-	
TAY	/	/	-	-	-	-	

Função	Bandeiras do Status Register					
	N	Z	C	I	D	V
TSX	/	/	-	-	-	-
TXA	/	/	-	-	-	-
TXS	-	-	-	-	-	-
TYA	/	/	-	-	-	-

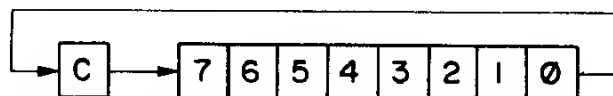
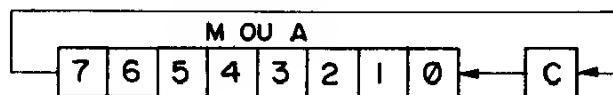
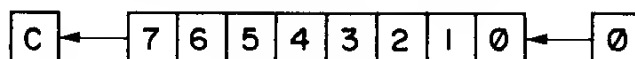
CÓDIGOS DAS OPERAÇÕES

00 - BRK	2F - Não usado
01 - ORA - (Indireto,X)	30 - BMI
02 - Não usado	31 - AND - (Indireto),Y
03 - Não usado	32 - Não usado
04 - Não usado	33 - Não usado
05 - ORA - Página Zero	34 - Não usado
06 - ASL - Página Zero	35 - AND - Página Zero,X
07 - Não usado	36 - ROL - Página Zero,X
08 - PHP	37 - Não usado
09 - ORA - Imediato	38 - SEC
0A - ASL - Acumulador	39 - AND - Absoluto,Y
0B - Não usado	3A - Não usado
0C - Não usado	3B - Não usado
0D - ORA - Absoluto	3C - Não usado
0E - ASL - Absoluto	3D - AND - Absoluto,X
0F - Não usado	3E - ROL - Absoluto,X
10 - BPL	3F - Não usado
11 - ORA - (Indireto),Y	40 - RTI
12 - Não usado	41 - EOR - (Indireto,X)
13 - Não usado	42 - Não usado
14 - Não usado	43 - Não usado
15 - ORA - Página Zero,X	44 - Não usado
16 - ASL - Página Zero,X	45 - EOR - Página Zero
17 - Não usado	46 - LSR - Página Zero
18 - CLC	47 - Não usado
19 - ORA - Absoluto, Y	48 - PHA
1A - Não usado	49 - EOR - Imediato
1B - Não usado	4A - LSR - Acumulador
1C - Não usado	4B - Não usado
1D - ORA - Absoluto,X	4C - JMP - Absoluto
1E - ASL - Absoluto,X	4D - EOR - Absoluto
1F - Não usado	4E - LSR - Absoluto
20 - JSR	4F - Não usado
21 - AND - (Indireto,X)	50 - BVC
22 - Não usado	51 - EOR - (Indireto),Y
23 - Não usado	52 - Não usado
24 - BIT - Página Zero	53 - Não usado
25 - AND - Página Zero	54 - Não usado
26 - ROL - Página Zero	55 - EOR - Página Zero,X
27 - Não usado	56 - LSR - Página Zero,X
28 - PLP	57 - Não usado
29 - AND - Imediato	58 - CLI
2A - ROL - Acumulador	59 - EOR - Absoluto,Y
2B - Não usado	5A - Não usado
2C - BIT - Absoluto	5B - Não usado
2D - AND - Absoluto	5C - Não usado
2E - ROL - Absoluto	5D - EOR - Absoluto,X

5E - LSR - Absoluto,X
 5F - Não usado
 60 - RTS
 61 - ADC - (Indireto,X)
 62 - Não usado
 63 - Não usado
 64 - Não usado
 65 - ADC - Página Zero
 66 - ROR - Página Zero
 67 - Não usado
 68 - PLA
 69 - ADC - Imediato
 6A - ROR - Acumulador
 6B - Não usado
 6C - JMP - Indireto
 6D - ADC - Absoluto
 6E - ROR - Absoluto
 6F - Não usado
 70 - BVS
 71 - ADC - (Indireto),Y
 72 - Não usado
 73 - Não usado
 74 - Não usado
 75 - ADC - Página Zero,X
 76 - ROR - Página Zero,X
 77 - Não usado
 78 - SEI
 79 - ADC - Absoluto,Y
 7A - Não usado
 7B - Não usado
 7C - Não usado
 7D - ADC - Absoluto,X
 7E - ROR - Absoluto,X
 7F - Não usado
 80 - Não usado
 81 - STA - (Indireto,X)
 82 - Não usado
 83 - Não usado
 84 - STY - Página Zero
 85 - STA - Página Zero
 86 - STX - Página Zero
 87 - Não usado
 88 - DEY
 89 - Não usado
 8A - TXA
 8B - Não usado
 8C - STY - Absoluto

8D - STA - Absoluto
 8E - STX - Absoluto
 8F - Não usado
 90 - BCC
 91 - STA (Indireto),Y
 92 - Não usado
 93 - Não usado
 94 - STY - Página Zero,X
 95 - STA - Página Zero,X
 96 - STX - Página Zero,Y
 97 - Não usado
 98 - TYA
 99 - STA - Absoluto,Y
 9A - TXS
 9B - Não usado
 9C - Não usado
 9D - STA - Absoluto,X
 9E - Não usado
 9F - Não usado
 A0 - LDY - Imediato
 A1 - LDA - (Indireto,X)
 A2 - LDX - Imediato
 A3 - Não usado
 A4 - LDY - Página Zero
 A5 - LDA - Página Zero
 A6 - LDX - Página Zero
 A7 - Não usado
 A8 - TAY
 A9 - LDA - Imediato
 AA - TAX
 AB - Não usado
 AC - LDY - Absoluto
 AD - LDA - Absoluto
 AE - LDX - Absoluto
 AF - Não usado
 B0 - BCS
 B1 - LDA - (Indireto),Y
 B2 - Não usado
 B3 - Não usado
 B4 - LDY - Página Zero,X
 B5 - LDA - Página Zero,X
 B6 - LDX - Página Zero,Y
 B7 - Não usado
 B8 - CLV
 B9 - LDA - Absoluto,Y
 BA - TSX
 BB - Não usado

BC -- LDY -- Absoluto,X	DE -- DEC -- Absoluto,X
BD -- LDA -- Absoluto,X	DF -- Não usado
BE -- LDX -- Absoluto,Y	E0 -- CPX -- Imediato
BF -- Não usado	E1 -- SBC -- (Indireto,X)
C0 -- CPY -- Imediato	E2 -- Não usado
C1 -- CMP -- (Indireto,X)	E3 -- Não usado
C2 -- Não usado	E4 -- CPX -- Página Zero
C3 -- Não usado	E5 -- SBC -- Página Zero
C4 -- CPY -- Página Zero	E6 -- INC -- Página Zero
C5 -- CMP -- Página Zero	E7 -- Não usado
C6 -- DEC -- Página Zero	E8 -- INX
C7 -- Não usado	E9 -- SBC -- Imediato
C8 -- INY	EA -- NOP
C9 -- CMP -- Imediato	EB -- Não usado
CA -- DEX	EC -- CPX -- Absoluto
CB -- Não usado	ED -- SBC -- Absoluto
CC -- CPY -- Absoluto	EE -- INC -- Absoluto
CD -- CMP -- Absoluto	EF -- Não usado
CE -- DEC -- Absoluto	F0 -- BEQ
CF -- Não usado	F1 -- SBC -- (Indireto),Y
D0 -- BNE	F2 -- Não usado
D1 -- CMP -- (Indireto),Y	F3 -- Não usado
D2 -- Não usado	F4 -- Não usado
D3 -- Não usado	F5 -- SBC -- Página Zero,X
D4 -- Não usado	F6 -- INC -- Página Zero,X
D5 -- CMP -- Página Zero,X	F7 -- Não usado
D6 -- DEC -- Página Zero,X	F8 -- SED
D7 -- Não usado	F9 -- SBC -- Absoluto,Y
D8 -- CLD	FA -- Não usado
D9 -- CMP -- Absoluto,Y	FB -- Não usado
DA -- Não usado	FC -- Não usado
DB -- Não usado	FD -- SBC -- Absoluto,X
DC -- Não usado	FE -- INC -- Absoluto,X
DD -- CMP -- Absoluto,X	FF -- Não usado



APÊNDICE B

DIFERENÇAS ENTRE O TK-2000 E O APPLE II

Existe um certo grau de compatibilidade entre o TK-2000 COLOR e o computador APPLE II PLUS e similares nacionais. Neste apêndice serão destacadas as diferenças mais importantes.

a) Diferença nos comandos BASIC

	TK-2000	APPLE II	Comandos no TK-2000 com o mesmo código do APPLE II
FLASH	-	S	SOUND
IN#	-	S	ASS
PR#	-	S	DSK
MA	S	--	
MP	S	--	
ASS	S	--	
LM	S	--	
TK2000	S	--	
MOTOR	S	--	
SOUND	S	--	
GR,HGR	(limpa só a janela)	(limpa a tela)	

b) Carregando fita de APPLE II no TK-2000

Para carregar as fitas em APPLESOFT BASIC no TK-2000, digite o comando MP e a tela mostrará um padrão de barras similar àquele que aparece ao ligar o aparelho.

Nesta posição pode-se também digitar o comando HOME para limpar a tela tornando agora possível carregar um programa gerado num APPLE II ou compatível no TK-2000 COLOR através do comando LOAD.

Se o programa não executar corretamente, deve-se procurar a solução para o erro observando o mapa da memória, ou conferindo no programa se não existem chamadas de subrotinas da ROM que possuem endereços diferentes no TK-2000 e no APPLE II ou similar.

Caso ao executar o programa, este seja interrompido com a mensagem de ERRO DE SINTAXE, provavelmente este ocorreu devido a um comando (FLASH, IN# ou PR#) que existia no APPLE (incompatível ao TK-2000), e deverá ser removido ou substituído apropriadamente de acordo ao contexto.

c) Diferenças no uso da RAM ocupado pelo vídeo

	TK-2000	APPLE II
TEXT0 página 1	2000-3FFFH	400-7FFH
TEXT0 página 2	A000-BFFFH	800-BFFH
BAIXA RES pág 1	2000-3FFFH	400-7FFH
BAIXA RES pág 2	A000-BFFFH	800-BFFH
ALTA RES pág 1	2000-3FFFH	2000-3FFFH
ALTA RES pág 2	A000-BFFFH	4000-5FFFH

d) Referência a programação em linguagem de máquina

- 1) Uso da página zero: Idêntico uso, exceto as posições "6", "7", "8" e "9" que são usadas como buffer temporários no TK-2000. "21H" representa o comprimento da janela de vídeo no APPLE II e no TK-2000 a margem direita da janela.
- 2) O uso da área de texto (end \$400H-7FFH): algumas partes desta área são usadas para parâmetros do sistema e dos periféricos. O TK-2000 não permite o display num programa em linguagem de máquina que escreva diretamente na área de memória em questão.
- 3) Leitura do teclado: No APPLE II a leitura do teclado é efetuada pela instrução LDA \$C000H e no TK-2000 deve ser usado uma rotina especial para simular a função. A rotina de reconhecimento e leitura do teclado está localizado a partir do endereço \$F043H da ROM do TK-2000.

Conversão de programas em linguagem de máquina para o TK-2000

Não há uma regra de ouro para converter programas em linguagem de máquina, mas as diferenças resultam, na grande maioria dos casos no tratamento do teclado, vídeo e mapeamento de memória, além de diferenças na ROM. Cada caso deve ser analisado individualmente, mas para simplificar, aqui temos algumas dicas:

1. Teclado

Encontre a instrução LDA \$C000 e substitua-a pela instrução JSR Keb, sendo que Keb é a seguinte rotina:

```
LDA $26
PHA
LDA $27
PHA
LDA $06
PHA
LDA $07
PHA
LDA $08
PHA
LDA $09
PHA
JSR $F043
PHA
TXA
PHA
TSX
TXA
EOR #$80
TAX
PLA
STA $0100,X
INX
PLA
STA $0100,X
PLA
STA $09
PLA
STA $08
PLA
STA $07
PLA
STA $06
PLA
STA $27
PLA
STA $26
LDA $0100,X
PHA
DEX
LDA $0100,X
TAX
PLA
RTS
```

Esta rotina deve ser colocada numa área livre da memória RAM. Uma opção mais simples é substituir LDA \$C000 por JSR \$F043, porém deve se saber que o conteúdo dos endereços 6, 7, 8, 9, 26 e 27 da página zero serão alterados, entretanto, na primeira opção eles são preservados.

2. Página 2 de Alta Resolução.

Encontre todas as instruções que se referem à página 2 no APPLE II (\$4000-5FFFH) e substitua-as pelo endereço correspondente no TK-2000 (\$A000-BFFFH).

3. Display Modo Texto

Encontre as instruções usadas para display no modo texto no APPLE II:

```
LDA  #C1
STA  $0401
```

e substitua-a pelo seguinte programa:

```
LDA  #01    ; EIXO X
STA  $24
LDA  #00    ; EIXO Y
STA  $25
LDA  #C1
JSR  $FDF0
```


APÊNDICE C

ENDEREÇAMENTO DAS LINHAS DE VÍDEO

Página 1

LINHA LINHA DE

AL.RES.	TEXTO	END. HEXADECIMAL	END.DECIMAL
0		\$2000-2027	8192-8231
1		\$2400-2427	9216-9255
2		\$2800-2827	10240-10279
3		\$2C00-2C27	11264-11303
4	0	\$3000-3027	12288-12327
5		\$3400-3427	13312-13351
6		\$3800-3827	14336-14375
7		\$3C00-3C27	15360-15399
8		\$2080-20A7	8320-8359
9		\$2480-24A7	9344-9383
10		\$2880-28A7	10368-10407
11		\$2C80-2CA7	11392-11431
12	1	\$3080-30A7	12416-12455
13		\$3480-34A7	13440-13479
14		\$3880-38A7	14464-14503
15		\$3C80-3CA7	15488-15527
16		\$2100-2127	8448-8487
17		\$2500-2527	9472-9511
18		\$2900-2927	10496-10535
19		\$2D00-2D27	11520-11559
20	2	\$3100-3127	12544-12583
21		\$3500-3527	13568-13607
22		\$3900-3927	14592-14631
23		\$3D00-3D27	15616-15655
24		\$2180-21A7	8576-8615
25		\$2580-25A7	9600-9639
26		\$2980-29A7	10624-10663
27		\$2D80-2DA7	11648-11687
28	3	\$3180-31A7	12672-12711
29		\$3580-35A7	13696-13735
30		\$3980-39A7	14720-14759
31		\$3D80-3DA7	15744-15783
32		\$2200-2227	8704-8743
33		\$2600-2627	9728-9767
34		\$2A00-2A27	10752-10791
35		\$2E00-2E27	11776-11815
36	4	\$3200-3227	12800-12839
37		\$3600-3627	13824-13863
38		\$3A00-3A27	14848-14887
39		\$3E00-3E27	15872-15911
40		\$2280-22A7	8832-8871
41		\$2680-26A7	9856-9895

42		\$2A80-2AA7	10880-10919
43		\$2E80-2EA7	11904-11943
44	5	\$3280-32A7	12928-12967
45		\$3680-36A7	13952-13991
46		\$3A80-3AA7	14976-15015
47		\$3E80-3EA7	16000-16039
48		\$2300-2327	8960-8999
49		\$2700-2727	9984-10023
50		\$2B00-2B27	11008-11047
51		\$2F00-2F27	12032-12071
52	6	\$3300-3327	13056-13095
53		\$3700-3727	14080-14119
54		\$3B00-3B27	15104-15143
55		\$3F00-2F27	16128-16167
56		\$2380-23A7	9088-9127
57		\$2780-27A7	10112-10151
58		\$2B80-2BA7	11136-11175
59		\$2F80-2FA7	12160-12199
60	7	\$3380-33A7	13184-13223
61		\$3780-37A7	14208-14247
62		\$3B80-3BA7	15232-15271
63		\$3F80-3FA7	16256-16295
64		\$2028-204F	8232-8271
65		\$2428-244F	9256-9295
66		\$2828-284F	10280-10319
67		\$2CA8-2C4F	11304-11343
68	8	\$3028-304F	12328-12367
69		\$3428-344F	13352-13391
70		\$3828-384F	14376-14415
71		\$3C28-3C4F	15400-15439
72		\$20A8-20CF	8360-8399
73		\$24A8-24CF	9384-9423
74		\$28A8-28CF	10408-10447
75		\$2CA8-2CCF	11432-11471
76	9	\$30A8-30CF	12456-12495
77		\$34A8-34CF	13480-13519
78		\$38A8-38CF	14504-14543
79		\$3CA8-3CCF	15538-15567
80		\$2128-214F	8488-8527
81		\$2528-254F	9512-9551
82		\$2928-294F	10536-10575
83		\$2D28-2D4F	11560-11599
84	10	\$3128-314F	12584-12623
85		\$3528-254F	13608-13647
86		\$3928-394F	14632-14671
87		\$3D28-3D4F	15656-15695
88		\$21A8-21CF	8616-8655
89		\$25A8-25CF	9640-9679

90		\$29A8-29CF	10664-10703
91		\$2DA8-2DCF	11688-11727
92	11	\$31A8-31CF	12712-12751
93		\$35A8-35CF	13736-13775
94		\$39A8-39CF	14760-14799
95		\$3DA8-3DCF	15784-15823
96		\$2228-224F	8744-8783
97		\$2628-264F	9768-9935
98		\$2A28-2A4F	10792-10831
99		\$2EA8-2E4F	11816-11855
100	12	\$3228-324F	12840-12879
101		\$3628-364F	13864-13903
102		\$3A28-3A4F	14888-14927
103		\$3E28-3E4F	15912-15951
104		\$22A8-22CF	8872-8911
105		\$26A8-26CF	9896-9935
106		\$2AA8-2ACF	10920-10959
107		\$2EA8-2ECF	11944-11983
108	13	\$32A8-32CF	12968-19007
109		\$36A8-36CF	13992-14031
110		\$3AA8-3ACF	15016-15055
111		\$3EA8-3ECF	16040-16079
112		\$2328-234F	9000-9039
113		\$2728-274F	10024-10063
114		\$2B28-2B4F	11048-11087
115		\$2F28-2F4F	12073-12111
116	14	\$3328-334F	13096-13135
117		\$3728-374F	14120-14159
118		\$3B28-3B4F	15144-15183
119		\$3F28-3F4F	16168-16207
120		\$23A8-23CF	9128-9167
121		\$27A8-27CF	10152-10191
122		\$2BA8-2BCF	11176-11215
123		\$2FA8-2FCF	12200-12239
124	15	\$33A8-33CF	13224-13263
125		\$37A8-37CF	14248-14287
126		\$3BA8-3BCF	15272-15311
127		\$3FA8-3FCF	16296-16335
128		\$2050-2077	8272-8311
129		\$2450-2477	9296-9335
130		\$2850-2877	10320-10359
131		\$2C50-2C77	11344-11383
132	16	\$3050-3077	12368-12407
133		\$3450-3477	13392-13431
134		\$3850-3877	14416-14455
135		\$3C50-3C77	15440-15479
136		\$20D0-20F7	8400-8439
137		\$24D0-24F7	9424-9463

138		\$28D0-28F7	10448-10487
139		\$2CD0-2CF7	11472-11511
140	17	\$30D0-30F7	12496-12535
141		\$34D0-34F7	13520-13559
142		\$38D0-38F7	14544-14583
143		\$3CD0-3CF7	15568-15607
144		\$2150-2177	8528-8567
145		\$2550-2577	9552-9591
146		\$2950-2977	10576-10615
147		\$2D50-2D77	11600-11639
148	18	\$3150-3177	12624-12663
149		\$3550-3577	13648-13687
150		\$3950-3977	14672-14711
151		\$3D50-3D77	15696-15735
152		\$21D0-21F7	8656-8695
153		\$25D0-25F7	9680-9719
154		\$29D0-29F7	10704-10743
155		\$2DD0-2DF7	11728-11767
156	19	\$31D0-31F7	12752-12791
157		\$35D0-35F7	13776-13815
158		\$39D0-38F7	14800-14839
159		\$3DD0-3DF7	15824-15863
160		\$2250-2277	8784-8823
161		\$2650-2677	9808-9847
162		\$2A50-2A77	10832-10871
163		\$2E50-2E77	11856-11895
164	20	\$3250-3277	12880-12919
165		\$3650-3677	13904-13943
166		\$3A50-3A77	14928-14967
167		\$3E50-3E77	15952-15991
168		\$22D0-22F7	8912-8951
169		\$26D0-26F7	9936-9975
170		\$2AD0-2AF7	10960-10999
171		\$2ED0-2EF7	11984-12023
172	21	\$32D0-32F7	13008-13047
173		\$36D0-36F7	14032-14071
174		\$3AD0-3AF7	15056-15095
175		\$3ED0-3EF7	16080-16119
176		\$2350-2377	9040-9079
177		\$2750-2777	10064-10103
178		\$2B50-2B77	11088-11127
179		\$2F50-2F77	12112-12151
180	22	\$3350-3377	13136-13175
181		\$3750-3777	14160-14199
182		\$3B50-3B77	15184-15223
183		\$3F50-3F77	16208-16247
184		\$23D0-23F7	9168-9207
185		\$27D0-27F7	10192-10231

186		\$2BD0-2BF7	11216-11255
187		\$2FD0-2FF7	12240-12279
188	23	\$33D0-33F7	13264-13303
189		\$37D0-37F7	14288-14327
190		\$3BD0-3BF7	15312-15351
191		\$3FD0-3FF7	16336-16375

LINHA AL.RES.	LINHA DE TEXTO	END. HEXADECIMAL	END.DECIMAL
0		\$A000-A027	40960-40999
1		\$A400-A427	41984-42023
2		\$A800-A827	43008-43047
3		\$AC00-AC27	44032-44071
4	0	\$B000-B027	45056-45095
5		\$B400-B427	46080-46119
6		\$B800-B827	47104-47143
7		\$BC00-BC27	48128-48167
8		\$A080-A0A7	41088-41127
9		\$A480-A4A7	42112-42151
10		\$A880-A8A7	43136-43175
11		\$AC80-ACA7	44160-44199
12	1	\$B080-B0A7	45184-45223
13		\$B480-B4A7	46208-46247
14		\$B880-B8A7	47232-47271
15		\$BC80-BCA7	48256-48295
16		\$A100-A127	41216-41255
17		\$A500-A527	42240-42279
18		\$A900-A927	43264-43303
19		\$AD00-AD27	44288-44327
20	2	\$B100-B127	45312-45351
21		\$B500-B527	46336-46375
22		\$B900-B927	47360-47399
23		\$BD00-BD27	48384-48423
24		\$A180-A1A7	41344-41511
25		\$A580-A5A7	42368-42407
26		\$A980-A9A7	43392-43431
27		\$AD80-ADA7	44416-44455
28	3	\$B180-B1A7	45440-45479
29		\$B580-B5A7	46464-46503
30		\$B980-B9A7	47488-47527
31		\$BD80-BDA7	48512-48551
32		\$A200-A227	41472-41511
33		\$A600-A627	42496-42535
34		\$AA00-AA27	43520-43559
35		\$AE00-AE27	44544-44583
36	4	\$B200-B227	45568-45607
37		\$B600-B627	46592-46631
38		\$BA00-BA27	47616-47655
39		\$BE00-BE27	48640-48679
40		\$A280-A2A7	41600-41639
41		\$A680-A6A7	42624-42663
42		\$AA80-AAA7	43648-43687
43		\$AE80-AEA7	44672-44711
44	5	\$B280-B2A7	45696-45735
45		\$B680-B6A7	46720-46759
46		\$BA80-BAA7	47744-47783

47		\$BE80-BEA7	48768-48807
48		\$A300-A327	41728-41767
49		\$A700-A727	42752-42791
50		\$AB00-AB27	43776-43815
51		\$AF00-AF27	44800-44839
52	6	\$B300-B327	45824-45863
53		\$B700-B727	46848-46887
54		\$BB00-BB27	47872-47911
55		\$BF00-BF27	48896-48935
56		\$A380-A3A7	41856-41895
57		\$A780-A7A7	42880-42919
58		\$AB80-ABA7	43904-43943
59		\$AF80-AFA7	44928-44967
60	7	\$B380-B3A7	45952-45991
61		\$B780-B7A7	46976-47015
62		\$BB80-BBA7	48000-48039
63		\$BF80-BFA7	49024-49063
64		\$A028-A04F	41000-41039
65		\$A428-A44F	42024-42063
66		\$A828-A84F	43048-43087
67		\$ACA8-AC4F	44072-44111
68	8	\$B028-B04F	45096-45135
69		\$B428-B44F	46120-46159
70		\$B828-B84F	47144-47183
71		\$BC28-BC4F	48168-48207
72		\$A0A8-A0CF	41128-41167
73		\$A4A8-A4CF	42152-42191
74		\$A8A8-A8CF	43176-43215
75		\$ACA8-ACCF	44200-44239
76	9	\$B0A8-B0CF	45224-45263
77		\$B4A8-B4CF	46248-46287
78		\$B8A8-B8CF	47272-47311
79		\$BCA8-BCCF	48296-48335
80		\$A128-A14F	41256-41295
81		\$A528-A54F	42280-42319
82		\$A928-A94F	43304-43343
83		\$AD28-AD4F	44328-44367
84	10	\$B128-B14F	45352-45391
85		\$B528-B54F	46376-46415
86		\$B928-B94F	47400-47439
87		\$BD28-BD4F	48424-48463
88		\$A1A8-A1CF	41384-41423
89		\$A5A8-A5CF	42408-42447
90		\$A9A8-A9CF	43432-43471
91		\$ADA8-ADCF	44456-44495
92	11	\$B1A8-B1CF	45480-45519
93		\$B5A8-B5CF	46504-46543
94		\$B9A8-B9CF	47528-47567
95		\$BDA8-BDCF	48552-48591

96		\$A228-A24F	41512-41551
97		\$A628-A64F	42536-42575
98		\$AA28-AA4F	43560-43599
99		\$AEA8-AE4F	44584-44623
100	12	\$B228-B24F	45608-45647
101		\$B628-B64F	46632-46671
102		\$BA28-BA4F	47656-47695
103		\$BE28-BE4F	48680-48719
104		\$A2A8-A2CF	41640-41679
105		\$A6A8-A6CF	42664-42703
106		\$AAA8-AACF	43688-43727
107		\$AEA8-AECF	44712-44751
108	13	\$B2A8-B2CF	45736-45775
109		\$B6A8-B6CF	46760-46799
110		\$BAA8-BACF	47784-47823
111		\$BEA8-BECF	48808-48847
112		\$A328-A34F	41768-41807
113		\$A728-A74F	42792-42831
114		\$AB28-AB4F	43816-43855
115		\$AF28-AF4F	44840-44879
116	14	\$B328-B34F	45864-45903
117		\$B728-B74F	46888-46927
118		\$BB28-BB4F	47912-47951
119		\$BF28-BF4F	48936-48975
120		\$A3A8-A3CF	41896-41935
121		\$A7A8-A7CF	42920-42959
122		\$ABA8-ABCF	43944-43983
123		\$AFA8-AFCF	44968-45007
124	15	\$B3A8-B3CF	45992-46031
125		\$B7A8-B7CF	47016-47055
126		\$BBA8-BBCF	48040-48079
127		\$BFA8-BFCF	49064-49103
128		\$A050-A077	41040-41079
129		\$A450-A477	42064-42103
130		\$A850-A877	43088-43127
131		\$AC50-AC77	44112-44151
132	16	\$B050-B077	45136-45175
133		\$B450-B477	46160-46199
134		\$B850-B877	47184-47223
135		\$BC50-BC77	48208-48247
136		\$A0D0-A0F7	41168-41207
137		\$A4D0-A4F7	42192-42231
138		\$A8D0-A8F7	43216-43255
139		\$ACD0-ACF7	44240-44279
140	17	\$B0D0-B0F7	45264-45303
141		\$B4D0-B4F7	46288-46327
142		\$B8D0-B8F7	47312-47351
143		\$BCD0-BCF7	48336-48375

144		\$A150-A177	41296-41335
145		\$A550-A577	42320-42359
146		\$A950-A977	43344-43383
147		\$AD50-AD77	44368-44407
148	18	\$B150-B177	45392-45431
149		\$B550-B577	46416-46455
150		\$B950-B977	47440-47479
151		\$BD50-BD77	48464-48503
152		\$A1D0-A1F7	41424-41463
153		\$A5D0-A5F7	42448-42487
154		\$A9D0-A9F7	43472-43511
155		\$ADD0-ADF7	44496-44535
156	19	\$B1D0-B1F7	45520-45559
157		\$B5D0-B5F7	46544-46583
158		\$B9D0-B8F7	47568-47607
159		\$BDD0-BDF7	48592-48631
160		\$A250-A277	41552-41591
161		\$A650-A677	42576-42615
162		\$AA50-AA77	43600-43639
163		\$AE50-AE77	44624-44663
164	20	\$B250-B277	45648-45687
165		\$B650-B677	46672-46711
166		\$BA50-BA77	47696-47735
167		\$BE50-BE77	48720-48759
168		\$A2D0-A2F7	41680-41719
169		\$A6D0-A6F7	42576-42615
170		\$AAD0-AAF7	43728-43767
171		\$AED0-AEF7	44752-44791
172	21	\$B2D0-B2F7	45776-45815
173		\$B6D0-B6F7	46800-46839
174		\$BAD0-BAF7	47824-47863
175		\$BED0-BEF7	48848-48887
176		\$A350-A377	41808-41847
177		\$A750-A777	42832-42871
178		\$AB50-AB77	43856-43895
179		\$AF50-AF77	44880-44919
180	22	\$B350-B377	45904-45943
181		\$B750-B777	46928-46967
182		\$BB50-BB77	47952-47991
183		\$BF50-BF77	48976-49015
184		\$A3D0-A3F7	41936-41975
185		\$A7D0-A7F7	42960-42999
186		\$ABD0-ABF7	43984-44023
187		\$AFD0-AFF7	45008-45047
188	23	\$B3D0-B3F7	46032-46071
189		\$B7D0-B7F7	47056-47095
190		\$BBD0-BBF7	49104-49143
191		\$BFD0-BFF7	49104-49143

APÊNDICE D ROTINAS ÚTEIS

Serão apresentadas algumas rotinas contidas na memória ROM que podem ser utilizadas pelo usuário. Para isto devem ser previamente carregados os devidos registradores do 6502 de acordo com as necessidades e executar com um JSR\$ no endereço indicado. No caso de se necessitar o acesso pelo modo BASIC, bastará aplicar um comando CALL acompanhado pelo endereço da rotina convertido em decimal.

\$FDED COUT Imprime um caracter

COUT é a subrotina standard de impressão de caracteres no dispositivo de saída. O caracter a ser impresso deve estar no acumulador. Esta função não altera com os valores contidos nos registradores. COUT chama a subrotina de impressão indicada por CSW (posições \$36 e \$37), normalmente para COUT1 (veja a seguir).

\$FDF0 COUT1 Imprime no vídeo

COUT1 imprime o caracter contido no acumulador na tela do TK-2000 a partir da atual posição do cursor, avançando o cursor em uma posição para cada caracter impresso. COUT1 permite controlar ainda os caracteres de controle RETURN, LF (alimentação de linha) e BELL (beep). Ao final da execução, retorna com os registradores intactos.

\$FE80 SETINV Converte vídeo para o modo inverso

SETINV altera o modo de atuação de COUT1. Todos os caracteres impressos na tela passarão a ser projetados em pontos pretos sobre fundo branco. Neste comando o registrador Y será alterado para \$3F, os restantes permanecerão intactos.

\$FE84 SETNORM Retorna o vídeo para o modo normal

SETNORM retorna o modo de atuação de COUT1 para projetar pontos brancos numa tela de fundo preto. O único registrador a ser alterado com esta função será o Y que passará a conter \$FF.

\$FD8E CROUT Gera um RETURN

CROUT envia um comando RETURN para o atual dispositivo de saída de dados.

\$FD8B CROUT1 Limpa o fim da linha e gera um RETURN

CROUT1 limpa toda a linha a partir da posição do cursor

e então chama CROUT.

\$FDDA **PRBYTE** Imprime um byte hexadecimal

Esta subrotina imprime o conteúdo do acumulador em hexadecimal no dispositivo atual de saída. Após a execução deste comando, o valor do acumulador será alterado.

\$FDE3 **PRHEX** Imprime um dígito hexadecimal

Esta subrotina imprime a metade menos significativa do byte contido no acumulador sob a forma de um dígito hexadecimal. Após a execução deste comando, o conteúdo do acumulador será alterado.

\$F941 **PRNTAX** Imprime A e X em hexadecimal

Imprime o conteúdo dos registros A e X sob a forma de uma forma de 4 dígitos hexadecimais. Assim, o conteúdo do acumulador será impresso no primeiro byte e o conteúdo do registro X será em segundo (AAXX). O conteúdo do acumulador geralmente será alterado após esta operação.

\$F948 **PRBLNK** Imprime 3 espaços

Manda 3 espaços brancos para o dispositivo de saída atual. Normalmente após esta operação, o acumulador conterá \$A0 e o registro X conterá 0.

\$F94A **PRBL2** Imprime espaços em branco

Esta subrotina imprime de 1 a 256 espaços brancos para o dispositivo de saída atual. O valor contido no registrador X irá especificar quantos espaços em branco serão impressos (variando de 1 a 256). Caso X=00 então o comando PRBL2 irá imprimir 256 brancos.

\$FF3A **BELL** Gera um caracter de "beep"

Esta subrotina manda um caracter de "beep" (CONTROL-G) para o dispositivo de saída. Após a execução o valor do acumulador passará a ser \$87. (Note que caso este dispositivo de saída seja uma impressora dotada de campainha, está soará ao imprimir tal caracter).

\$FBDD **BEEL1** Gera um caracter de "beep" na TV

Esta subrotina irá emitir um "beep" de de 1 KHz durante 0.1 segundos apenas no alto-falante da TV conectada ao TK-2000 (ou monitor de vídeo dotada de alto-falante).

\$FD6A GETLN Manda uma linha com o símbolo de pronto ">"

GETLN é a subrotina que permite a entrada de linha de dados ou de programa. Os programas podem chamar GETLN com o próprio carácter de pronto na localização \$33; GETLN irá retornar com a linha de entrada no endereço de entrada do buffer (começando no endereço \$200) e o registro X armazenará o comprimento da linha de entrada (que no máximo poderá ser de 256 caracteres).

\$FD67 GETLNZ Obtém a linha de entrada

GETLNZ é um ponto de entrada alternativo para GETLN que provoca um carriage return para o dispositivo de saída antes de cair num GETLN.

\$FD6F GETLN1 Obtém linha de entrada sem simb. de "pronto"

GETLN1 é um ponto de entrada alternativo para GETLN que não provoca o símbolo de "pronto" antes de iniciar a linha de entrada de dados. Se, no entanto, o usuário cancelar a linha de entrada, ou com alguns espaços do cursor para trás ou apenas com um CONTROL-X, então GETLN1 irá emitir o conteúdo do endereço \$33 com um sinal de pronto quando passar para outra linha.

\$FCAB WAIT Atraso

Esta subrotina provoca um atraso específico de tempo, quando então retorna para o programa de origem. O total de atraso é especificado através do conteúdo do acumulador. Com o conteúdo do acumulador sendo A, o atraso será de $(26 + 27*A + 5*A^2)$ microsegundos. WAIT retorna então com o registrador A igual a zero e os registradores X e Y intactos.

\$F864 SETCOL Determina a cor para gráficos em Baixa RES.

O comando SETCOL é utilizada para plotar gráficos em baixa resolução pois determina a cor de acordo com o valor contido no acumulador (00-0F).

\$F85F NEXTCOL Incrementa a cor de gráficos em baixa res.

Esta subrotina incrementa de #3 no conteúdo atual de color que é utilizada num gráfico em baixa resolução.

\$F800 **PLOT2** Plota um ponto em baixa resolução

Esta subrotina plota um ponto (bloco) simples de cor já predeterminada em uma tela de baixa resolução. A posição vertical está determinada no acumulador e a horizontal no registrador Y. Este comando apenas irá modificar o conteúdo do acumulador, os outros registros permanecerão intocados.

\$F819 **HLINE** Desenha uma linha horizontal de blocos

Esta subrotina desenha uma linha horizontal de blocos de cor predeterminada em baixa resolução. Pode-se chamar HLINE com a coordenada vertical contida no acumulador, a coordenada da extremidade esquerda da linha está contida no registrador Y, e o da extremidade direita está contida no endereço \$2C. HLINE alterará o conteúdo do acumulador e também do registrador Y.

\$F828 **VLINE** Desenha uma linha vertical de blocos

Idêntico à subrotina anterior, somente que plota uma linha vertical com a coordenada horizontal contida no registro Y, a coordenada do topo da linha contida no acumulador, e a coordenada de baixo contida na posição \$2D. VLINE irá retornar com somente o valor do acumulador alterado.

\$F832 **CLRSCR** Limpa a tela inteira da baixa resolução

CLRSCR limpa a tela inteira do vídeo de baixa resolução inclusive a parte destinada ao modo texto. Este comando altera o conteúdo do acumulador e do registrador Y.

\$F836 **CLRTOP** Limpa somente o vídeo de baixa res.

CLRTOP é o mesmo que CLRSCR, exceto que este limpa apenas a parte destinada aos gráficos de baixa resolução mantendo intacto a área de texto.

\$F869 **SCRN** Lê a tela de baixa resolução

Esta subrotina retorna com o número da cor de um simples bloco de baixa resolução. O modo de localização do bloco é o mesmo utilizada pela subrotina PLOT. A cor correspondente ao bloco retornará diretamente no acumulador. Nenhum outro registrador será alterado.

\$FF2D **PRERR** Imprime "ERR"

Emite a palavra "ERR", seguido do caracter "beep" (CONTROL-G), para o equipamento atual de saída. O conteúdo do

acumulador será alterado..

\$FF4A IOSAVE Salva todos os registradores

O conteúdo de todos os registradores internos do 6502 serão armazenados nas localizações de memória de \$7F0 até \$7F4 na ordem A-X-Y-P-S. Após isso o conteúdo do acumulador e registrador X serão alterados.

\$FF3F IORESTORE Carrega todos os registradores

Esta subrotina irá carregar os valores contidos nas posições \$7F1 até \$7F4 nos registradores X-Y-P-S respectivamente, e o conteúdo de \$45 no acumulador.

\$F043 SCAN1 Verifica 1 vez o teclado

Se Carry=0, indica que não foi pressionada tecla alguma, e se Carry=1, alguma tecla foi pressionada, e o código da tecla se encontra no acumulador..

APÊNDICE E - BIBLIOGRAFIA

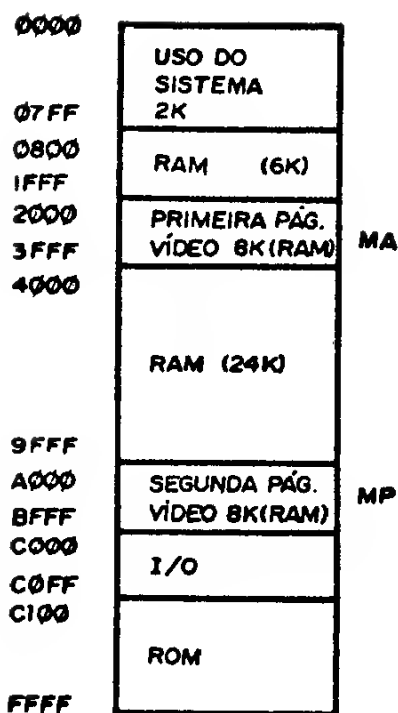
A seguir apresentamos a literatura que pode ser usada como referência para melhor compreensão do ser TK-2000:

- 1 - Synertek 6500 Hardware Manual
- 2 - 6502 Assembly Language Programming
Lance A. Leventhal
- 3 - 6502 Assembly Language Subroutines
Lance A. Leventhal

A literatura para o computador Apple pode ser utilizada como fonte de informação adicional, observando sempre as diferenças entre os equipamentos.

APÊNDICE F

MAPA DA MEMÓRIA



O mapa de memória mostra as diferentes funções a que foram dedicadas as várias áreas disponíveis.

Os primeiros 2k de RAM são de uso do sistema operacional e do BASIC do TK-2000 COLOR, assim como os últimos 16k são dedicados as portas de I/O (entrada e saída) e a ROM.

Podemos observar que sobraram entres os endereços 0800H e BFFFH duas áreas dedicadas a páginas de vídeo. A área restante é livre e disponível. As páginas de vídeo contém as informações que aparecem na imagem da sua TV ou monitor. Como só uma aparece por vez, pode-se selecionar qual delas se deseja no momento, pelo uso dos comandos MA e MP. MA seleciona a página compreendida entre 2000H e 3FFFH, e MP seleciona a página compreendida

entre A000H e BFFFH. Ao ligar seu computador é selecionada automaticamente a primeira página. Se desejar usar só uma página de vídeo, a área da outra fica livre e disponível para o usuário.

O programa em BASIC começa automaticamente a partir do endereço 0800H. Se o programa foi maior que 6k, existirá um conflito com a primeira página de vídeo. Neste caso existem duas opções: a) digite o comando MP e a área normalmente dedicada a primeira página de vídeo poderá ser usada para o programa, permitindo assim acesso a 38k de RAM. b) Defina a posição de início do seu programa a partir do endereço 4000. Isto é obtido digitando os seguintes comandos diretos:

```
>LOMEM: 16384
>POKE 103,1
>POKE 104,64
>POKE 16384,0
>NEW
```

O comando LOMEM: é usado para definir o limite inferior da área de variáveis. O comando HIMEM: define o limite superior. Por exemplo, após ligar o TK-2000 COLOR digite o seguinte comando direto:

```
HIMEM: 2060
```

Agora tente entrar vários comandos do tipo:

```
10 REM
20 REM
```


e aparecerá o comentário de erro: EXCEDE MEMORIA indicando que não há mais espaço na área de programa BASIC.

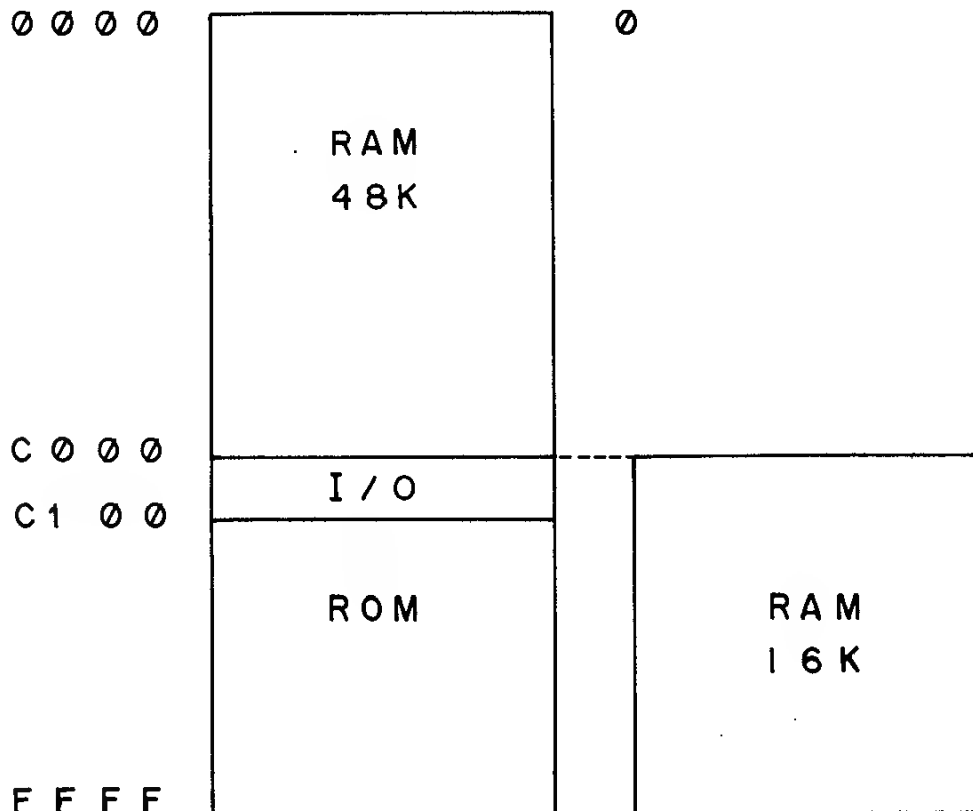
APENDICE G

TK-2000 II 64K

Em sua versão básica, o TK-2000 possui 48 Kbytes de RAM. No TK-2000 II versão com 64K, o sistema foi amplificado pela incorporação de 16K adicionais de memória RAM dinâmica. Tal expansão viabilizou a utilização daqueles programas aplicativos que exigem uma memória mais potente.

Essa memória adicional está localizada na mesma região ocupada pela ROM e pelos dispositivos de entrada-saída (I/O). É selecionada através de um soft-switch.

A figura abaixo esquematiza o mapeamento de memória do TK-2000 II 64K:



O sistema de decodificação interna do TK-2000 II 64K permite selecionar, através de dois registros de I/O (endereços \$C05A e \$C05B), a memória RAM ou ROM.

Quando você liga o TK-2000 II 64K, o registro RO (\$C05A) é automaticamente acionado. Assim, dos 64 K de RAM, normalmente são utilizados apenas 48 K, ficando os últimos 16 K desabilitados.

Para modificar esta situação, ou seja, habilitar todos os 64K da RAM e desabilitar a ROM, você deve acessar o registro RA (\$C05B). E para retornar à posição original, você deve acessar o endereço \$C05A.

Use um artifício de software, fazendo com que o programa residente na ROM seja transferido para a RAM, habilitando-a.

Experimente este programa:

```
>LM      (RETURN)
```

```
@6200:AD 00 C1 8E 5B C0 8D 00 C1 8E 5A C0 E
E 01 62 EE 07 62 D0 EC EE 02 62 EE 08 62
D0 E4 60      (RETURN)
```

Verifique se não há erros de digitação, acompanhando a listagem disassemblada do programa:

```
@6200L      (RETURN)
```

```
6200- AD 00 C1 LDA $C100
6203- 8E 5B C0 STX $C05B
6206- 8D 00 C1 STA $C100
6209- 8E 5A C0 STX $C05A
620C- EE 01 62 INC $6201
620F- EE 07 62 INC $6207
6212- D0 EC BNE $6200
6214- EE 02 62 INC $6202
6217- EE 08 62 INC $6208
621A- D0 E4 BNE $6200
621C- 60 RTS
```

E então, continue:

```
@6200G      (RETURN)      (transfere, para a RAM, o programa
                           contido na ROM)
```

```
@C05B      (RETURN)      (passa a RAM de expansão para o
                           primeiro plano e a ROM para o
                           segundo plano)
```

Agora você pode modificar o conteúdo de qualquer endereço de memória, pois seu computador está operando somente com a memória RAM.

Obs.: Tendo selecionado o registro RA, não pressione as teclas RESET, pois assim você voltará a acessar o registro RO.

A expansão de RAM pode ser usada para guardar informações. Por exemplo, num programa BASIC, que obviamente usa a ROM, deve-se observar o seguinte procedimento:

- 1 - O programa BASIC deverá passar controle a uma rotina em linguagem de máquina, gerenciadora do bloco e posicionada na área normal de RAM.
- 2 - A rotina deverá habilitar a RAM e desabilitar a ROM, acessando o registro RA (\$C05B).
- 3 - O bloco de RAM deverá ser acessado pela rotina gerenciadora, de acordo com a necessidade.
- 4 - Agora a rotina gerenciadora deverá habilitar a ROM e desabilitar a RAM, acessando o registro RD (\$C05A) e, posteriormente, retornar ao BASIC.

Experimente reprogramar um dos cinquenta caracteres gráficos:

Entre em linguagem de máquina:



>LM (RETURN)

E digite:

@ F400:01 03 05 09 11 21 41 FF (RETURN)

Com CONTROL-C e (RETURN), volte ao BASIC:

>■

Selecione o modo gráfico, pressionando CONTROL-B e (RETURN). Agora pressione CONTROL-SHIFT-1 e aparecerá no vídeo o símbolo  no lugar do  contido na ROM.

Nota: Os primeiros 256 bytes da expansão de 16K de RAM (de \$C000 a C0FF) não são disponíveis. Correspondem à região de I/O.

APENDICE H

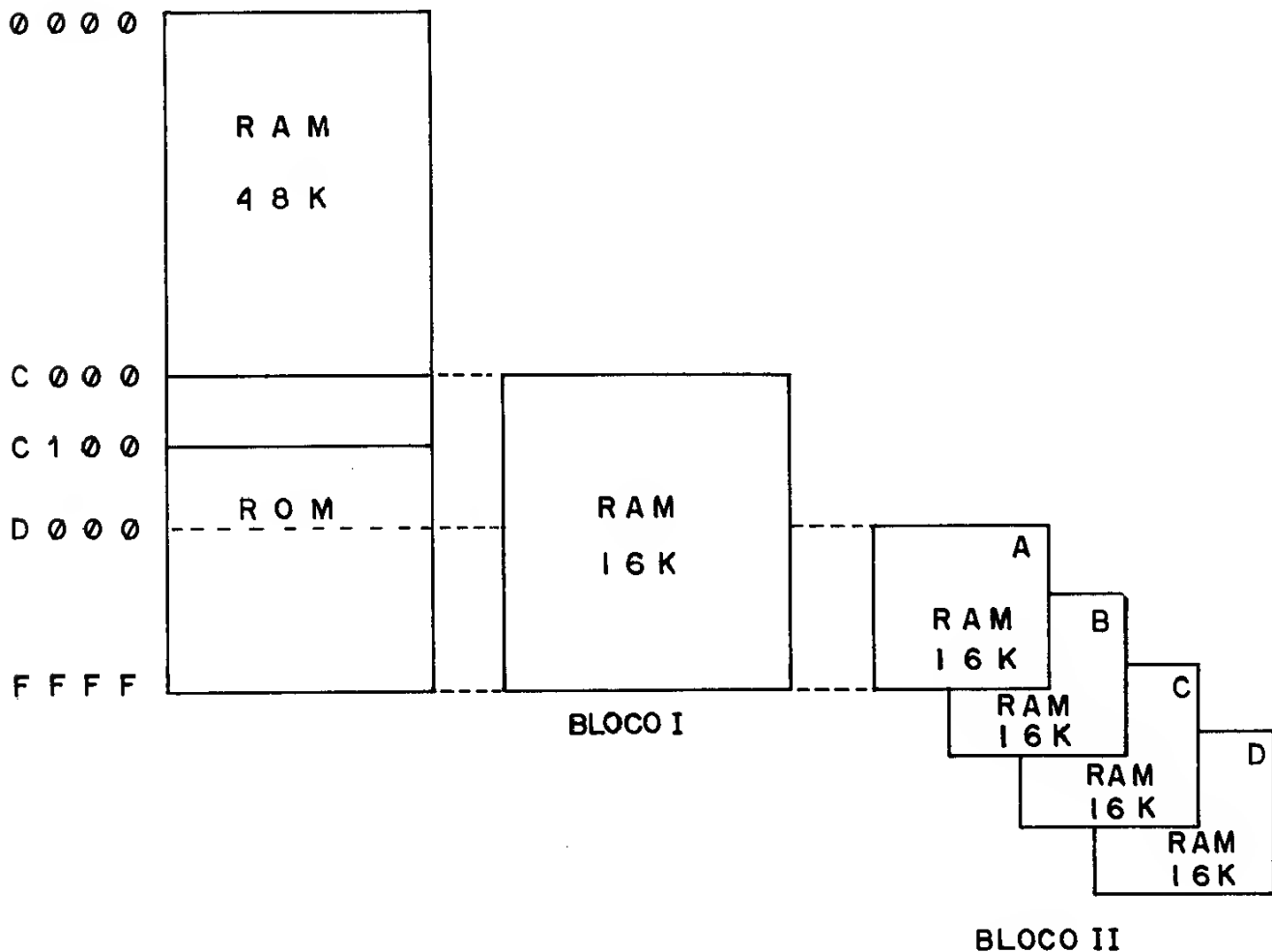
TK-2000 II 128K

O TK-2000 II 128K possui 128 Kbytes de memória de trabalho (RAM). Comparado com o TK-2000 na versão de 48K, tem capacidade de memória para armazenar o equivalente a 80K mais.

A expansão de memória foi projetada em dois blocos:

- um bloco de 16K, controlado por meio dos endereços de I/O \$C05A e \$C05B;
- outro bloco de 64K, controlado através de doze endereços de I/O compreendidos entre \$C080 e \$C08B.

Observe o mapeamento da memória, no esquema abaixo:



O endereço \$C05A do bloco I foi definido para mapear ROM e I/O na região situada entre \$C000 e \$FFFF. Enquanto o endereço \$C05B, também do bloco I, serve ao mapeamento de RAM na região que vai de \$C100 a \$FFFF.

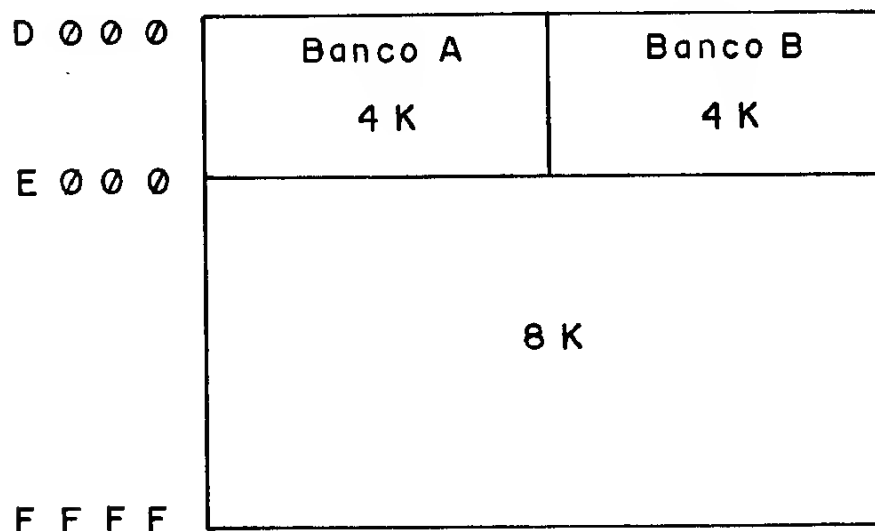
Obs.: Os primeiros 256 bytes da expansão não são utilizáveis, pois correspondem a endereços de I/O do TK-2000.

O gerenciamento do bloco II é diferente. A expansão de 64K foi distribuída em quatro bancos de 16K (bancos A, B, C e D), mapeados na região compreendida entre \$D000 e \$FFFF. Essa faixa de endereços corresponde a 12 Kbytes endereçados diretamente pela CPU.

Como havia 16K de RAM para endereçar e apenas 12 K de endereços, organizou-se cada um dos 16K em três bancos, da seguinte forma:

- . banco 1, com 8Kbytes, alocado entre \$E000 e \$FFFF;
- . banco 2, correspondente a dois bancos (banco a/ banco b), com 4Kbytes cada um, sobrepostos e alocados entre \$D000 e \$DFFF.

BLOCO DE 16 K



O gerenciamento do bloco II é indicada na tabela abaixo:

ENDEREÇO	MODO DE OPERAÇÃO			BANCO 2 (Banco de 4K)		BLOCO II (Blocos de 16 K)			
	RAM WRITE	RAM READ	MEMÓRIA PRINCIPAL	BANCO A	BANCO B	BLOCO A	BLOCO B	BLOCO C	BLOCO D
C 0 8 0			• (2)	•					
C 0 8 1	• (1)	•		•					
C 0 8 2		•		•					
C 0 8 3	• (1)		• (3)	•					
C 0 8 4						•			
C 0 8 5							•		
C 0 8 6								•	
C 0 8 7									•
C 0 8 8			• (2)		•				
C 0 8 9	• (1)	•			•				
C 0 8 A		•			•				
C 0 8 B	• (1)		• (3)		•				

Notas:

- (1) Obtém-se o Modo de Operação RAM Write fazendo dois acessos a um dos endereço indicados: \$C081, \$C083, \$C089 ou \$C08B. Se for feito apenas um acesso, será obtida a RAM Read tão-somente.
- (2) Seleciona-se a Memória Principal, para leitura e escrita, através de \$C080 ou \$C088.
- (3) Seleciona-se a Memória Principal, apenas para leitura, na região compreendida entre \$C100 e \$FFFF, através dos endereços \$C083 ou \$C08B.

Observações:

- 1 - Os endereços situados entre \$C080 e \$C083 são utilizáveis para selecionar o Modo de Operação e o Banco a, de 4K.
- 2 - Os endereços de \$C084 a \$C087 servem para selecionar um dos quatro blocos de 16K. Não influem na seleção do Modo de Operação nem do banco de 4K.
- 3 - Os endereços alocados entre \$C088 e \$C08B destinam-se a selecionar o Modo de Operação e o Banco b, de 4K.
- 4 - Os endereços de \$C08C a \$C08F são reservados às futuras expansões, portanto não devem ser utilizados no TK-2000 II 128K.
- 5 - Chama-se de Memória Principal aquela que se constitui do conjunto dos 15 3/4 K de ROM, 1/4 K de I/O, 48K de RAM e mais 80 K de RAM de expansão.

